NOT MEASUREMENT SENSITIVE

MIL-STD-2407
28 JUNE 1996
SUPERSEDING
MIL-STD-600006
27 APRIL 1996

DEPARTMENT OF DEFENSE

INTERFACE STANDARD FOR VECTOR PRODUCT FORMAT

AMSC N/A

AREA MCGT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

FOREWORD

1. This interface standard is approved for use by all Departments and Agencies of the Department of Defense.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to: DMA (ATISI), MS A-10, 8613 Lee Highway, Fairfax, Virginia 22031-2137, by using the Standardization Document Proposal (DD Form 1426) appearing at the end of this document or by letter.

1. SCOPE

1.1 Scope. The vector product format (VPF) is a standard format, structure, and organization for large geographic databases that are based on a georelational data model and are intended for direct use. VPF is designed to be compatible with a wide variety of applications and products. VPF allows application software to read data directly from computer-readable media without prior conversion to an intermediate form. VPF uses tables and indexes that permit direct access by spatial location and thematic content and is designed to be used with any digital geographic data in vector format that can be represented using nodes, edges, and faces. VPF defines the format of data objects, and the georelational data model provides a data organization within which software can manipulate the VPF data objects. A product specification corresponding to a specific database product determines the precise contents of feature tables and their relationships in the database. In this context, each separate product or application is defined by a product specification and implemented by using VPF structures.

1.2 Applicability. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

2. APPLICABLE DOCUMENTS

2.1 General. The documents listed in this section are needed to meet the requirements specified in sections 3, 4, and 5 of this specification. This section does not include documents cited in other sections of this specification or recommended for additional information as examples. While every effort has been made to ensure the completeness of this list, document users are cautioned that they must meet all requirements documents cited in sections 3, 4, and 5 of this specification, whether or not they are listed.

2.2 Government documents.

2.2.1 Specifications, standards, and handbooks. The following specifications, standards, and handbooks form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those listed in the issue of the Department of Defense Index of Specifications and

Standards (DODISS) and supplement thereto, cited in the solicitation (see 6.2).

HANDBOOKS

DEPARTMENT OF DEFENSE

MIL-HDBK-850 -	DoD Glossary of Mapping, Charting and Geodesy (MC&G) Terms

(Unless otherwise indicated, copies of federal and military specifications, standards, and handbooks are available from the Standardization Documents Order desk, Bldg. 4D, 700 Robbins Ave., Philadelphia, PA 19111-5094.)

2.2.2 Other Government documents, drawings, and publications. This section is not applicable to this standard.

2.3 Non-Government publications. The following documents form a part of this document to the extent specified herein. Unless otherwise specified, the issues of the documents which are DoD adopted are those listed in the issue of the DODISS cited in the solicitation. Unless otherwise specified, the issues of documents not listed in the DODISS are the issues of the documents cited in the solicitation (see 6.2).

INTERNATIONAL STANDARDS ORGANIZATION (ISO)

ISO 646	-	Information Processing—ISO 7-Bit Coded Character Set for Information Exchange.
ISO 2022	-	Information Processing—ISO 7-Bit and 8-Bit Coded Character Sets—Code Extension Techniques.
ISO 2375	-	Data Processing—Procedure for Registration of Escape Sequences.
ISO 6937	-	Information Processing—Coded Character Sets for Text Communication.
ISO 8601	-	Data Elements and Interchange Formats - Information Interchange—Representation of Dates and Times.
ISO 8859.1	-	Information processing - 8-Bit single byte coded graphic character sets - Part 1: Latin Alphabet No. 1

ISO 9660	- Information Processing–Volume and File Structure of CD-ROM for Information Interchange.
ISO 10646-1	- Information Technology - Universal Multiple-Octet Coded Character Set (USC), Part 1: Architecture and Basic Multilingual Plane

AMERICAN NATIONAL STANDARDS INSTITUTE/INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (ANSI/IEEE)

ANSI/IEEE 754-1986	- IEEE Standard for Binary Floating Point Arithmetic.
ANSI X3.4-1977	- Code for Information Interchange (ASCII) Adopted in FIPSPUB 1-1,

(Copies of ISO and ANSI documents are available from the American National Standards Institute, 1430 Broadway, New York, NY 10018.)

FEDERAL INFORMATION PROCESSING STANDARDS (FIPS)

FIPSPUB 151-1	- POSIX: Portable Operating System Interface for Computer Environments.

(Copies of Federal Information Processing Standards (FIPS) are available to Department of Defense activities from the Standardization Document Order Desk, 700 Robbins Avenue, Building 4D, Philadelphia, PA 19111-5094. Others must request copies of FIPS from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161-2171.)

NORTH ATLANTIC TREATY ORGANIZATION - STANDARDIZATION AGREEMENT (NATO STANAG)

STANAG 7074	- Digital Geographic Information Exchange Standard (DIGEST), Version 1.2

Note: VPF products may reside on a variety of media. If a VPF database product resides on CD-ROM, ISO 9660 CD-ROM format is required for that product.

2.4 Order of precedence. In the event of a conflict between the text of this document and the references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

3. DEFINITIONS

Area feature. A geographic entity that encloses a region; for example, a lake, administrative area, or state.

Area feature class. A collection of area features that maintains a homogeneous set of attributes. Implies the use of face primitives.

Area feature table. The implementation of an area feature class in a VPF attribute table.

Attribute. A property of an entity; for example, the color of a building, the width of a road, or the accuracy level of a database. Defined subtypes of an attribute are the feature-attribute, coverage attribute, database attribute, and library attribute.

Attribute accuracy. Attribute accuracy refers to the accuracy or reliability of attribute data within the limits described by feature completeness. If attribute accuracy information is not available in the above form, a description of known attribute accuracy characteristics may be substituted.

Attribute completeness. Attribute completeness refers to the percentage of feature attribute fields not populated by null or default values.

Attribute table. A collection of identically formatted (defined) attribute rows. An attribute table inherits the properties of a VPF table, but also may have value description tables.

Attribute value. The specific value of an attribute; for example, green for building color, 48 feet for road width, level 2 for the accuracy of a database.

Byte order. A hardware implementation of an encoding scheme. It determines the order in which bytes are stored in a long word. Two commonly used orders are little-endian, or least significant first (i.e., 1234); and big-endian, or most significant first (i.e., 4321).

Cartographic primitive. A primitive with no topological relationship to adjoining or surrounding primitives. The text primitive is a cartographic primitive.

Code table. A set of character specifications. A code table defines the alpha numeric and special characters that are used in a computer system to model written languages.

Column. The set of all values of a particular attribute within a table.

Column type. The relational model uses column types to implement the data type of an attribute. For instance, the column ELEVATION could have an integer column type.

Complex feature. A single feature that relates directly to other features rather than to a primitive. A single feature composed of other features, either simple or complex.

Complex feature class. A feature class that includes two or more other feature classes (simple or complex).

Complex feature table. An implementation of a complex feature class in VPF.

Compound feature. A single simple feature composed of more than one primitive of the same type. A compound feature may cross tile boundaries.

Compound key. A group of columns used together to create a key in a relational table.

Connected node. One of the two node primitive types. It is used to represent linked features that are zero dimensional at a particular scale. Connected nodes are always found at the ends of edges and are topologically linked to the edges. Connected nodes are used in two ways: (1) to define edges topologically (always) and (2) to represent point features that are found at a juncture of linear features, such as overpasses, locks in a canal, or underground utility access points. Under the first usage, the connected nodes are referred to as start and end nodes. Under the second usage, attributes will be associated with the point features related to the connected nodes. If many edges intersect a node, only one edge will be maintained per node in the connected node table; other edges are linked by using winged-edge topology. All connected nodes which lie on a tile boundary will have cross-tile components (tile_id and first_edge).

Containing face. A face that contains one or more entity nodes. Used to establish the relationship from an entity node to its face, if level 3 topology is present.

Coordinate. A specified position in Cartesian space. The value takes the form of a short or long floating point value. Z values (if any) are ignored during the enforcement and use of planar graphs.

Coordinate array. A fixed-length list of coordinate tuples.

Coordinate pair. A specified position in a two-dimensional grid, where the first position relates to the X axis and the second position relates to the Y axis.

Coordinate string. A variable-length list of coordinate tuples.

Coordinate triplet. A specified position in a three-dimensional grid, where the first position relates to the X axis, the second position relates to the Y axis, and the third position relates to the Z axis.

Coordinate tuple. A coordinate pair or triplet.

Coverage. A set of feature classes that has a specified spatial extent and in which the primitives interconnect as described by the coverage's topology.

Coverage attribute. A property of a coverage. The coverage attribute table contains properties for all coverages in the library.

Cross-tile topology. The encoding of topological relationships in such a manner that those relations are maintained even when a coverage has been physically partitioned into multiple tiles.

Data dictionary. A collection of tables with entries that define the meaning of attributes and the allowable values (or ranges of values).

Data syntax. A description of the computer-readable (bit-level) representation of data.

Database. A collection of related libraries as defined by a product specification.

Database attribute. A property of a database.

Date status. Date status refers to the date at which the data was introduced or modified in the database. This date of entry is used as a proof of modification for a single data element and permits the statistical interpretation of groups of data elements.

Direct access. Retrieval of data by reference to its location on a storage medium rather than relative to the previously retrieved data. The access mechanism goes directly to the data in question. This access method is normally required for on-line data usage.

Directory. A file that contains a list of the unique names, beginning addresses, and lengths of other files.

Edge. A one-dimensional primitive used to represent the location of a linear feature and/or the borders of faces. Depending upon the level of topology, edges may be topologically linked to nodes, edges, and faces. Edges are composed of an ordered collection of two or more coordinate tuples (pairs or triplets). At least two of the coordinate tuples must be distinct. The orientation of an edge can be recognized by the ordering of the coordinate tuples.

Encapsulation. A set format that serves to identify data elements.

Encoding. The assignment of bit-patterns to data types in a computer. For example, one given bit arrangement may define an integer data type (e.g., 2's complement, 1's complement, or biased), whereas another may describe a character data type (e.g., ASCII, EBCDIC).

End node. The terminating node of an edge.

Entity. A general term for any object that is being modeled or defined within a database.

Entity node. One of the two node primitive types. It is used to represent isolated features that are zero dimensional at a particular scale. Entity nodes are topologically linked to a containing face when face topology is present. Entity nodes cannot occur on edges.

Escape sequence. A special character code used to extend the characters used in a character code table.

Face. A region enclosed by an edge or set of edges (a face has area). Faces are topologically linked to their surrounding edges as well as to the other faces that surround them. Faces are always non overlapping, exhausting the area of a plane.

Feature. A model of a real world geographic entity. A zero-, one-, or two-dimensional entity of uniform attribute scheme from an exhaustive attribute distribution across a plane, or a set of such entities sharing common attribute values. Simple and complex are types of features.

Feature attribute. A property of a feature.

Feature class. A set of features that shares a homogeneous set of attributes. A feature class consists of a set of tables that includes one or more primitive tables and one or more attribute tables. A feature class has the same columns of attribute information for each feature. Every feature class has one and only one feature table. The two types of feature classes are the simple feature class and complex feature class. The subtypes of the simple feature class are the point feature class, line feature class, area feature class, and text feature class.

Feature class attribute table. The feature class attribute table contains properties for all feature classes in a coverage. The feature class attribute table is required to support the feature index tables.

Feature class schema table. A table that stores the composition rules of each feature class. This table describes the definition for each feature class and the way in which each table in a feature class relates to other tables.

Feature completeness. Feature completeness refers to the degree to which all features of a given type for the area of the data set have been included.

Feature index table. The feature index table is constructed specifically to provide for rapid retrieval of feature information when given a selected primitive. It provides a join index from primitive to feature.

Feature join table. A table that identifies 1:N or N:1 relationships between features and other features or primitives. Simple features may be composed of one or more primitive instances, and complex features may be composed of one or more simple features or other complex features. A primitive instance may belong to more than one feature.

Feature table. A table made up of rows of features in a feature class. These rows collectively form the feature table for that feature class.

Field. A field contains a single attribute value of a single entity. Fields in a table identify the data types contained within each table.

File. A named stream of bytes. The three VPF file types are directory, table, and index file.

First edge. An edge arbitrarily selected as the first edge to enable traversing around a connected node.

Fixed field. A field made up of a predefined number of bytes. Fixed fields are generally used for numeric data, or when blank entries are significant.

Foreign key. One or more columns in one table that are used as a primary key in another table.

Geographic entity. A phenomenon characterized by its locational context and about which spatially referenced information is stored.

Geographic information system (GIS). An organized collection of computer hardware, software, geographic data, and standard operating procedures for efficiently capturing, storing, maintaining, retrieving, analyzing, displaying, and reporting spatially referenced information.

Geographic reference table. A table that defines the coordinate system of a library.

Geometric primitive. The basic geometric units of representation; specifically, nodes, edges, and faces.

Georelational data model. A generic conceptual model in which geographic information is represented by using a combination of vector geometry and planar, topology, and relational data models.

Index. A mechanism used to quickly identify a particular record or group of records based on a table's primary key.

Index file. The implementation of an index stored in a file. There are variable-length indexes, spatial indexes, thematic indexes, and feature indexes.

Inner ring. The inner boundary of a face, composed of edges ordered in a sequence. A face may have none or any arbitrary number of inner rings.

Integrated data. A geographic data set in which all feature data are contained in a single coverage. Opposite of layered data.

Key. In a relational data model, one or more columns (attributes) whose values uniquely identify or can be used to select a row.

Layered data. Feature data thematically separated into separate coverages. Opposite of integrated data.

Left edge. The left edge is the first neighbor of the current edge as one moves counterclockwise around the start node of the current edge.

Left face. The face to the left of an edge in a traverse from the start node to the end node.

Levels of topology. See topology.

Library. A collection of one or more coverages contained within a specified spatial extent, all of which share a single coordinate system. Coverages may be tiled in a library. All tiled coverages in a library must share a common tiling scheme, however.

Library attribute. A property of a library. The library attribute table describes the properties of each library in a database.

Line feature. A geographic entity that defines a linear (one-dimensional) structure; for example, a river, road, or a state boundary.

Line feature class. A collection of line features that maintains a homogeneous set of attributes. Composed of edge primitives.

Line feature table. The implementation of a line feature class in a VPF attribute table.

Lineage information. Information that describes processing tolerances, interpretation rules applied to source materials, and basic production and quality assurance procedures. Lineage information should include all available information from the source.

Logical consistency. Logical consistency refers to the fidelity of the relationships encoded in a data set. In a VPF data set, logical consistency requires that all topological foreign keys match the appropriate primitive, that all attribute foreign keys match the appropriate primitive or features, and that all tables described in the feature class schema tables maintain the relationships described.

Media volumes. As used herein, the number of distinct media that comprise a VPF database. For instance, there may be four CD-ROM media volumes in one database.

Medium. A data storage device (e.g., a CD-ROM, hard disk drive, magnetic tape, or floppy disk).

Metadata. Information about data; more specifically, information about the meaning of other data.

Minimum bounding rectangle. A rectangle of coordinate tuples that defines the minimum and maximum coordinates of an entity. Abbreviated as MBR.

Names reference coverage. A coverage that contains (at a minimum) a point feature table with columns indicating a place name and its known coordinate. Used to help a user locate places by name.

NaN. Stands for not a number. Used as a floating point null value in VPF.

Neutral format. A characteristic of a data model that does not contain product-specific information.

Node. A zero-dimensional geometric primitive that is composed of a single coordinate tuple (pair or triplet). There are two types of nodes: entity nodes and connected nodes. Only one node can occupy a single geographic location.

Outer ring. The outermost boundary of a face, composed of edges ordered in a sequence.

Pathname. A file name that uniquely identifies the location path to a file within a series of one or more directories.

Planar model. A planar model is a two-dimensional surface in which every point has a neighborhood (a two-dimensional region) that is topologically equal to a flat disk. It is implemented as a planar graph {N, E, F} with a finite number of nodes N = {n1, n2,...}, edges E = {e1, e2,...}, and faces F = {f1, f2,...} bounded by edges and nodes. Each edge has an orientation from its first (starting) coordinate tuple to its last coordinate tuple. Also, each face of the graph has a certain orientation (cycle) around its edges and nodes. Each edge of a planar model is incident with exactly two faces.

Point feature. A geographic entity that defines a zero-dimensional location; for example, a well or a building.

Point feature class. A collection of point features that maintains a homogeneous set of attributes. Composed of node primitives.

Point feature table. The implementation of a point feature class in a VPF attribute table.

Pointer. A field within a record or within an index that contains the address of a record.

Polygon. Thematically homogenous areas composed of one or more faces.

Positional accuracy. Positional accuracy refers to the root mean square error (RMSE) of the coordinates relative to the position of the real world entity being modeled. Positional accuracy shall be specified without relation to scale and shall contain all errors introduced by source documents, data capture, and data processing.

Primary key. A key whose value uniquely identifies a row.

Primitive. The smallest component of VPF, of which all features are composed. There are three geometric primitives (nodes, edges, faces) and one cartographic primitive (text).

Primitive table. A primitive table inherits the properties of a VPF table, but may also have an associated minimum bounding rectangle table and/or a spatial index file.

Product specification. A document that defines the precise content and format of a specific product. It contains technical requirements and database design decisions such as coding, tiling, special relationships between entities, and so forth. In the context of the VPF, each separate product or application is defined by a product specification and implemented by using VPF structures.

Right edge. The right edge is the first neighbor of the current edge as one moves counterclockwise around the end node of the current edge.

Right face. The face to the right of an edge in a traverse from the start node to the end node.

Ring. A connected set of edges that composes the face border. Any single ring is only referenced to and by a single face. If the same set of edges is shared by two different faces, two rings that correspond to the two faces are created from the single edge set. Rings only occur at level 3 topology (when faces are also present).

Row. An ordered collection of fields pertaining to the entity. A tuple in a relation.

Row id. An integer that uniquely identifies each row in a table.

Schema. A description (or picture or diagram) of the structures of a database system.

Schema table. A schema table defines the tables and their relationships within a coverage.

Semantics. The implied meaning of data. Used to define what entities mean with respect to their roles in a system.

Shape line. An ordered set of one or more coordinate tuples that define the placement and shape of a text primitive.

Simple feature class. Consists of a single type of primitive (face, edge, node, or text) and a feature table. There are four subtypes of simple feature classes: point, line, area, and text.

Source. Source information describes the origin or derivation of a single feature, primitive, or attribute. It includes information about processing of the data as well as information about the data source.

Spatial index. A data structure file that allows for the rapid identification of a primitive by using the values of the primitive's coordinates.

Start edge. An edge arbitrarily selected as the start edge to enable traversing through the ring table asociated with a face.

Start node. The first node of an edge.

Syntax. The rules governing the construction of a machine language or machine representation of entities.

Table. An organizational structure for data content. In the relational model, a table is a group of repeating rows defined by columns. Equivalent to a relation.

Text feature. A cartographic entity that relates a textual description to a zero- or one-dimensional location. A text feature usually contains information such as font, color, and height.

Text feature class. A collection of text features that maintains a homogenous set of attributes. Composed of text primitives.

Text feature table. The implementation of a text feature class in a VPF attribute table.

Text primitive. Characters placed in specific locations in a coordinate system. Text is a cartographic object, rather than a geographic entity, since it does not participate in topology. A text array indicates a fixed-length string of characters. A text string indicates a variable-length collection of characters.

Thematic attribute. A column in a table that provides a thematic description of a feature. For example, a feature class that contains rivers may have attributes such as width, depth, and name.

Thematic index. A file that allows software to access the row ids of its associated table. In a VPF table, the index is created on a column. Four special indexes are used for feature tables: point, line, area, and text thematic indexes.

Theme. An organizational concept used in the design of spatial databases. Common themes in spatial geographic databases are transportation, hydrology, and soil/land suitability.

Tile. A spatial partition of a coverage that shares the same set of feature classes with the same definitions as the coverage. The topology of each tile is independent of that of each other tile in the coverage.

Tiled coverage. A coverage that has been physically partitioned into tiles.

Tiling scheme. The scheme used to define tile shape and size and to identify tiles (assign identification numbers).

Topology. The branch of mathematics concerned with geometric relationships unaltered by elastic deformation. In geographic applications, topology refers to any relationship between connected geometric primitives that is not altered by continuous transformation. VPF recognizes four levels of topology. Level 0 topology manipulates the purely geometric aspects of the spatial data. No topological information is stored in level 0 topology. Level 1 topology maintains a nonplanar graph. Level 2 topology maintains a planar graph. Level 3 topology explicitly represents the faces defined by the planar graph. See Figure 10.

Traverse. An algorithm that uses winged-edge topology to retrieve a series of neighboring edges to satisfy a query of a network.

Triplet id. A variable-length structure used to contain information for crossing tile boundaries. The first field contains the internal primitive id (referred to as ID). The second field contains the tile reference coverage id (TILE_ID), and the third field contains the primitive id in the associated tile (EXT_ID).

Tuple. See coordinate tuple.

Universe face. The unbounded region surrounding a level 3 topology coverage. The universe face always maintains the first record in a face table.

Variable-length column. A column whose length is determined by the amount of storage needed to store its contents. Useful for character strings and coordinate strings.

Vector. Indicates a collection of coordinate tuples to define a geographic or geometric entity.

Vector product format. A standard format, structure, and organization for large geographic databases based on a georelational data model and intended for direct access. Abbreviated as VPF.

VPF table. A VPF table consists of a table header and rows of data sharing the same column definitions, each having a unique row id. Primitive tables, attribute tables and feature tables are all special purpose VPF tables.

Winged-edge topology. A topological construct that connects each edge to two of its neighboring edges, allowing topologic traversal of an edge and/or face network. A neighboring edge is any edge that shares a start or end node with the original edge. An edge has a start node, which is connected to the left edge, and an end node, which is connected to the right edge.

4. GENERAL REQUIREMENTS

4.1 General. Vector product format is a generic geographic data model designed to be used with any digital geographic data in vector format that can be represented using nodes, edges, and faces. VPF is based upon the georelational data model, combinatorial topology, and set theory (see appendix A for discussions of these concepts).

A VPF-compliant database product must include all mandatory tables and columns which are described in section 5. Similarly, a VPF-compliant application must properly interpret all mandatory and optional tables and columns enumerated in this document.

4.2 VPF characteristics. VPF is a model for digital geographic databases that are intended to allow flexibility in encoding and yet permit direct data access from a variety of applications operating on different computer systems. VPF characteristics are as follows:

a. Sheetless database support. VPF is designed to support a sheetless database by providing logically continuous topological relationships even when the database itself is physically partitioned into tiles. VPF structures support the query and retrieval of data that extends across tile boundaries.

b. Neutral format. VPF has a product-neutral format that must be used in combination with an individual product specification to create a product. VPF is topologically structured and supports various levels of topology, from a simple list of coordinates to planar model topology.

c. Attribute support. VPF uses tables for attribute handling. The tables support both simple and complex features.

d. Data dictionary. VPF contains a self-defining data dictionary that permits user understanding of features and their attributes.

e. Text and metadata support. Text information may be encoded either as attributes of features or as 'free floating' text primitives. VPF is compatible with all written languages, including those with accented characters and diacritical marks. In addition to supporting basic cartographic information, VPF supports a variety of metadata files containing information about all or part of the database.

f. Index file support. Index files that are desirable to enhance database retrieval performance are also incorporated within VPF. These can include spatial and thematic indexes.

g. Direct access. VPF allows application software to read data directly from the storage medium without prior conversion to another format. VPF uses tables and indexes that permit direct access by spatial location and thematic content.

h. Flexible, general-purpose schema. VPF can represent digital geographic data in vector format by providing flexibility in the modeling of any feature data organization, from fully layered to completely integrated. VPF supports coordinate pairs and triplets, multiple scales, and the creation of multiple products.

i. Data quality. VPF includes standards for data quality reporting and representation. It provides multiple methods for the representation of the spatial and aspatial aspects of data quality.

j. Feature definitions. VPF organizes features and thematic attributes to allow creation of products that are logically consistent and complete.

4.3 Relationship between VPF and specific products. VPF establishes a standard data model and organization, providing a consistent interface to data content. Data content itself shall be defined in a product specification that determines the content of the feature tables and the relationships between them. VPF can also accommodate additional tables that are not required by VPF itself. Without further standardization these additional tables, although VPF compliant in structure, will provide column names and attribute values which may not be understood by others. Use of tables outside of those described in this standard may limit interoperability. FIGURE 1 illustrates the relationship between VPF and specific products.

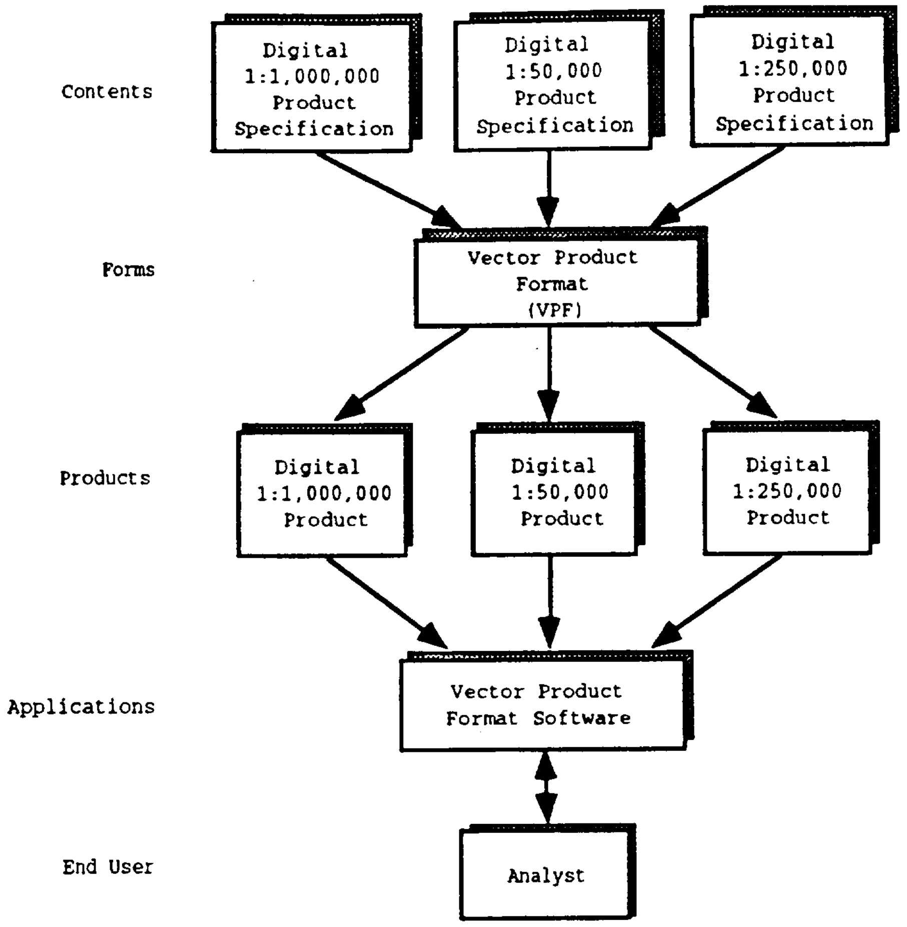

Contents
Digital 1:1,000,000 Product Specification
Digital 1:50,000 Product Specification
Digital 1:250,000 Product Specification
Forms
Vector Product Format (VPF)
Products
Digital 1:1,000,000 Product
Digital 1:50,000 Product
Digital 1:250,000 Product
Applications
Vector Product Format Software
End User
Analyst

FIGURE 1. Relationship between VPF and specific products.

4.4 VPF hierarchy. VPF can be viewed as a five-level hierarchy of definitions (FIGURE 2) that increase in degree of abstraction from the bottom up. The bottom two levels define the physical representations of various data structures utilized in VPF. The data structure level concerns the logical representation of VPF data objects. These objects are elements in the VPF data model. The data model describes the data objects and the relationships among them. The top level, which contains the product specification, is used to tailor VPF to the requirements of the product.

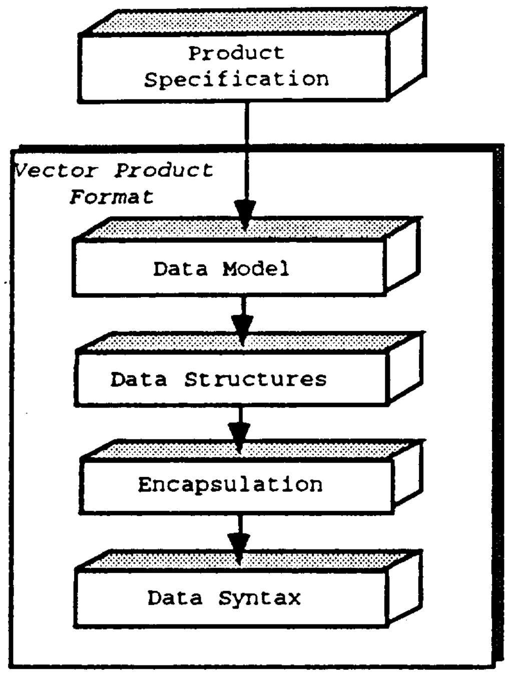


FIGURE 2. Vector product format structure.

This document gives a definition of VPF that encompasses the bottom four levels. A product specification is a combination of the conceptual database design and implementation details required to develop a product that is compliant with VPF. The conceptual modeling of feature classes and coverages is the first responsibility of the author of the product specification. This includes defining the list of features, attributes, attribute values, and providing their definitions. The physical design of the product database is also the responsibility of the author of the product specification. Physical design considerations include determining the tiling scheme, the topology level, the feature to/from primitive relationships (i.e., 1:1, 1:N, N:1, and N:M), the column types, and the table definitions.

Vector product format is defined in section 5. Section 5.2 defines the data objects in VPF and the various roles these objects play in the data model. The logical data structures are described in section 5.3, which explains the implementation of VPF; that is, how to construct VPF-compliant databases and applications.

Encapsulation is defined in section 5.4, which identifies the data structure fields. Data syntax is described in section 5.5.

5. DETAILED REQUIREMENTS

5.1 General. This section describes the necessary components of vector product format. Section 5.2 discusses the conceptual components of the VPF data model (see appendix A for an overview describing the data model). Section 5.3 contains definitions of the data structures implemented in VPF. The encapsulation of VPF field types, table construction, and indexing structures are discussed in section 5.4. The encoding of the data syntax is found in section 5.5.

5.2 VPF data model. The discussion of the data model is broken into three subsections: data organization, VPF data model components, and data quality. Each of these subsections addresses the data model from a different perspective. The data organization subsection (5.2.1) addresses VPF by defining the physical structures that make it up. Only three structures are used to implement the entire VPF data model: directories, tables, and indexes. The data model component subsection (5.2.2) addresses VPF by defining the entities in a geographic database and describing the way in which these entities are captured through the physical VPF structures, starting with the most basic components (primitives) and continuing through the other levels (features, coverages, libraries, and database). Finally, the data quality subsection (5.2.3) describes the options available in VPF for the maintenance of data quality information at any level in the data model.

5.2.1 Data organization. All VPF data are organized in the form of files. A file is a named, sequentially ordered stream of bytes (FIGURE 3). Files may be created, deleted, opened, closed, read (from byte m to byte n), and written (from byte m to byte n).

byte 1
byte 2
byte 3
.....
byte n
.....

FIGURE 3. Byte stream.

VPF uses only three types of files: directories, tables, and indexes. All directory and file names in VPF data bases are to be in lower case. Within this document, upper case letters have been used for readability.

5.2.1.1 Directory. The directory is a file that identifies the names of a collection of files, and their beginning addresses and lengths (TABLE 1).

TABLE 1. Directory structure.

Directory		
Name	Address	Length
File Name	Location on Medium	Length In Media Storage Units

VPF directories are strictly hierarchical; each file is contained in exactly one directory. File names must be unique within a directory. A file referenced by a directory is said to be contained in that directory. A file contained in a directory may be referenced by a special form of its name called a pathname (because it contains the location path to the file). A pathname has the following form:

<directory name><separator><file name>

where:
< > indicates that the enclosed name element is to be replaced with the actual text string indicated.

VPF uses the backslash character (\) as the generic pathname directory separator. For platforms requiring a different separator, software will replace the backslash with the appropriate separator character. For example, if a file named ROADS.LFT is contained in a directory named URBANAREAS, and the separator is (\), the resulting pathname would be:

URBANAREAS\ROADS.LFT

Directories are themselves files, so they may be contained in other directories. They are referenced by pathname the same way as other files. Thus, if URBANAREAS is contained in LIBRARY1, the resulting pathname would be:

LIBRARY1\URBANAREAS

Finally, pathnames may be combined. Since the directory that contains a file can be contained within a directory itself, it is necessary to have a form of file name that uniquely identifies that file contained within that directory (file names, while unique within a directory, are not unique between directories). For our example, that form would be:

LIBRARY1\URBANAREAS\ROADS.LFT

5.2.1.2 Tables. In the VPF data model, the table is the organizational structure for all data content. All tables in a VPF database share a common basic structure; this structure, which is described in the VPF table components section below (5.2.1.3), is mandatory for all VPF tables.

By definition, a VPF table must include at least the basic structure. Optionally, a VPF table can also reference additional structures: the narrative table, thematic index(s), column narrative table(s) and value description table(s). A table can also have an associated variable length index and a spatial index (for primitive tables). In the VPF data model, all geographic phenomena are modeled by VPF tables or by tables derived from a VPF table. A table derived from a VPF table is one that possesses all the properties of a VPF table but also has additional properties that support other specific functions.

The primitive table and the attribute table are examples of derived VPF tables. A derived table can also be further specialized to satisfy a particular need. A feature table may be derived from the attribute table, for example.

A VPF table may have an associated index file for variable-length records and a narrative table. A primitive table (discussed in section 5.2.2.1) may possess these two tables associated with its VPF table, but may also have a spatial index file and a minimum bounding rectangle table. An attribute table may also possess the tables associated with a VPF table, but may also have value description tables that provide the data dictionary for the table. The same data dictionary table may be shared by more than one attribute table. Finally, a feature table inherits the tables associated with an attribute table, but may also have a thematic index file. Feature tables are discussed further in section 5.3.3.1.

5.2.1.3 VPF table components. VPF tables consist of the following parts: a table header, a row identifier, and the table contents (under special situations (section 5.2.2.3.3) a table may contain only a header). The table header contains the metadata about a table and the column definitions. Columns are defined by a name and a data type; each column must have a name that is unique within the table.

Data contents in VPF tables are organized into rows and columns. All rows in a table share the same column definitions. Each row in the table is defined by a unique row identifier (row id). The row ids shall start at 1 and be sequential with no gaps in the numbering. TABLE 2 depicts the principal components of a VPF table.

TABLE 2. VPF table structure.

Table Header Metadata and column definitions:	
a. Table description b. Narrative table name (optional) c. Column definitions: Column name Field type Field length Key type Column textual description Optional value description table name Optional thematic index name Optional column narrative table name	
ID	**Table Contents**
Indicates the starting position of each row.	The data composing the table that match the column definitions.

This document describes the column definitions for all the VPF standard-specified columns, and the table organization for those columns. No specific ordering of columns within a table is required. Product specifications may require a specific column order. Data columns and tables described in this document are labeled either mandatory or optional. A VPF product must include all mandatory tables and columns. It is not possible to remove any mandatory column from any table. A VPF-compliant application must be able to process a VPF product and interpret all mandatory and optional columns as described in this document.

Additional product-specific columns are allowed by VPF. If present, these columns must be defined in their product specifications. Product-specific columns must not alter the use of the columns specified in this document.

5.2.1.4 Indexes. A table may have associated indexes. If a table contains a variable-length coordinate string column or a variable-length text string column, a separate index file must be present.

In addition to variable-length indexes, VPF also supports spatial and thematic indexes. Spatial indexes contain references to row data that are based on the value of a coordinate column. Thematic indexes contain references to row data that are based on the value of noncoordinate columns.

5.2.1.5 Narrative tables Each VPF table may have an associated narrative table that provides miscellaneous information about the VPF table. The purpose of the narrative table is to provide the database designer with the ability to record comments or information pertinent to the associated table. The narrative table name is stored in the VPF table's header information. In addition, VPF provides for optional narrative tables keyed to individual columns within a table. The narrative table name is stored as the third optional entry in the column definition (see TABLE 2).

5.2.1.6 Attribute tables. Real-world objects are referred to as entities or features; they are modeled in tables in VPF. The properties of entities are called attributes. In an attribute table, one table column is defined for each attribute describing an object. Each object occupies a row in the table. Examples of attributes include data quality, size, and name. A sample attribute table is shown in TABLE 3.

A column or a group of columns that can be used to identify or select a row is called a key. A unique key is a key that uniquely identifies each row. One unique key is designated the primary key; each table has one and only one primary key. In the city attribute table (TABLE 3), the Built-Up Area column is the primary key.

TABLE 3. City attribute table.

ID	**Built-Up Area**	**State**	**Population Size**	**Median Income per Household**
implicit	*character string*	*character string*	*binary integer*	*binary integer*
UNIQUE KEY	PRIMARY KEY	NON-UNIQUE	NON-UNIQUE	NON-UNIQUE
1	Los Angeles	California	2966850	15735
2	New York	New York	7071639	13854
3	Salt Lake City	Utah	163033	13211
4	Las Vegas	Nevada	164674	17468
5	San Francisco	California	1366383	16782

A relational join is a database operation that brings together a number of tables into a new relation by using a set of common keys. The tables in such joins are called base tables. When a common key in a join is the primary key in one of the base tables but not in another, the non-primary (yet common) key is called a foreign key. In the city attribute table (TABLE 3), the State column is a foreign key; in the state attribute table (TABLE 4), the State column is the primary key. In the city attribute table (TABLE 3) the State column becomes a foreign key only through its reference by the state attribute table (TABLE 4).

TABLE 4. State attribute table.

ID	**State**	**Area (sq. mi.)**	**Total Population**
implicit	*character string*	*binary integer*	*binary integer*
UNIQUE KEY	PRIMARY KEY	NON-UNIQUE	NON-UNIQUE
1	California	158706	26365000
2	Nevada	110561	936000
3	New York	49108	17783000
4	Utah	84899	1645000

5.2.2 VPF data model components. The VPF data model may be considered to be layered into four structural levels (FIGURE 4). At the lowest level, a VPF database consists of feature classes. In the database, these feature classes are defined using VPF primitive and attribute tables. Feature classes make up coverages, which in turn make up libraries; and finally, a database is made up of libraries.

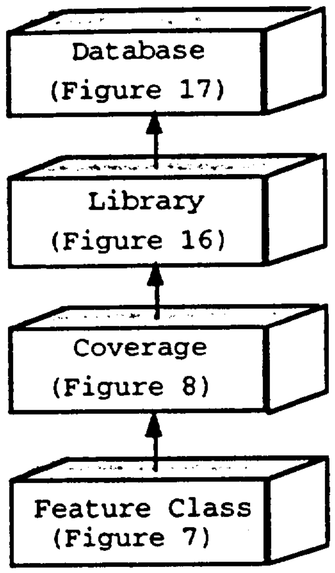


FIGURE 4. VPF structural levels.

An analogy can be drawn between VPF and written language. Letters are at the bottom of the language hierarchy. Words are made up of letters. In turn, sentences are made up of words. An essay is created from sentences, and a collection is made up of essays. Each of these entities has a distinct and different meaning not possessed by the entities below. The content of each entity, however, depends on that of the constituent entities. Databases and libraries are used primarily to facilitate data access, whereas coverages (which incorporate topology) are used to define the relationships between features.

5.2.2.1 Primitives. There are three geometric primitives in VPF: nodes, edges, and faces (FIGURE 5). As FIGURE 5 shows, there are two types of node primitives: entity nodes and connected nodes. There is one type of cartographic primitive, text. These four primitives are combined to model any geographic phenomena using vector geometry. All primitives except text can be linked to each other by topological relationships, which are discussed further in section 5.2.2.3.1.

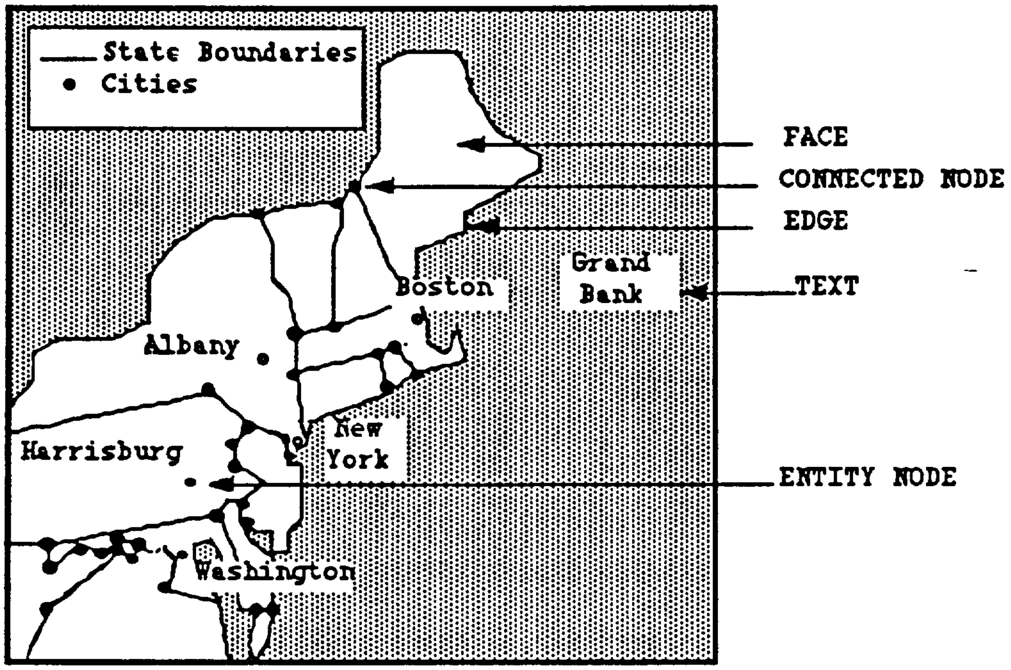
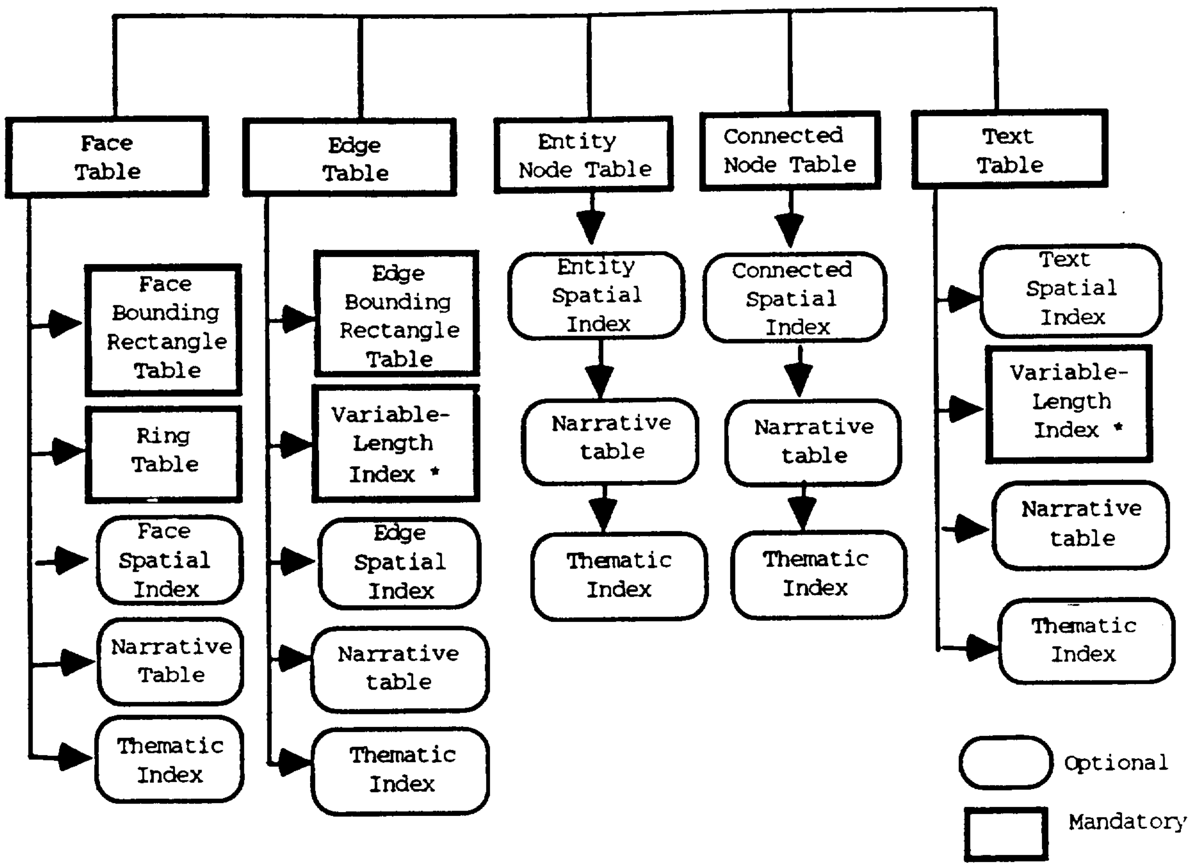


FIGURE 5. Geometric and cartographic primitives.



* mandatory when variable length column is defined in a table

FIGURE 6. Primitive directory contents.

The following sections summarize each of the primitives. FIGURE 6 depicts each primitive and its associated tables and indexes.

5.2.2.1.1 Nodes. Nodes are zero-dimensional primitives that are used to store significant locations. No two nodes can occupy the same coordinate tuple. There are two types of nodes: entity nodes and connected nodes.

a. Entity nodes. Entity nodes are used to represent isolated features that are either truly zero dimensional, such as survey points, or too small to resolve at the collection scale, such as water towers at 1:24,000 scale. An entity node is topologically linked to its containing face when face topology is present. Entity nodes cannot fall on an edge.

b. Connected nodes. Connected nodes are always found at the ends of edges and are topologically linked to the edges. Connected nodes are used in two ways: (1) to define edges topologically and (2) to represent point features that are found at the start or end of an edge of linear features, such as overpasses, locks in a canal, or underground utility access points. Under the first usage, the connected nodes are referred to as start and end nodes. Under the second usage, attributes will be associated with the point features related to the connected nodes. All connected nodes are included in the connected node table. If many edges intersect a node, only one edge will be maintained per node in the connected node table; other edges are linked by using winged-edge topology (APPENDIX B). All connected nodes which lie on a tile boundary will have cross-tile components (tile_id and first_edge).

5.2.2.1.2 Edges. Edges are one-dimensional primitives that are used to represent the locations of linear features (such as roads) and the borders of faces. Edges are composed of an ordered collection of two or more coordinate tuples (pairs or triplets). At least two of the coordinate tuples must be distinct. The orientation of an edge can be recognized by the ordering of the coordinate tuples.

Edges are topologically defined by nodes at ends (levels 1-3 topology); edges, in turn, define faces (level 3 topology). In addition to the Start Node and End Node columns, the edge primitive table contains column information (Right Edge, Left Edge, Right Face, Left Face) that is necessary to support higher levels of topology. This topology information permits the query and retrieval of features. The direction of an edge is its orientation from start node to end node. Each edge table has an associated edge bounding rectangle (EBR) table which contains the minimum bounding rectangle (MBR) for each edge. There is a one to one relationship between the edge table and its associated edge bounding rectangle table. Appendix B describes the use of winged-edge topology, which is used with edge primitives.

5.2.2.1.3 Faces. A face is a two-dimensional primitive enclosed by edges; faces are used to represent area features, such as countries, inland water, or urban areas. Faces are defined by topological references to a set of edges that compose the face border. A face may have interior borders as well as exterior borders, allowing for faces that have other smaller faces within them. This relation consists of a reference to the start of a closed ring of edges, which may then be followed clockwise to close the ring. A face may consist of multiple rings; there may be one outer ring and zero or more inner rings. Faces are non-overlapping, and the faces in a coverage completely exhaust the area of a plane. Each face table has an associated face bounding rectangle (FBR) table which contains the minimum bounding rectangle for each face. There is a one to one relationship between the face table and its associated face bounding rectangle table.

5.2.2.1.4 Text. Text is a cartographic rather than a geometric object. Text strings can be placed in specific locations in geographic space. Text can be used to associate names with regions that are vague or ill defined, such as the Rocky Mountains. A text primitive may also be used when the name of a feature needs to be located in a specific relationship to a feature and could not otherwise be reproduced. For example, the text "Pacific Ocean" may be required for graphic display on a map, and may therefore be encoded as a text string, even though it is also being stored as an attribute of a face in a hydrographic coverage. Text primitives do not participate in topology.

5.2.2.2 Feature classes. Features are defined using primitive and attribute tables by means of relational modeling. Tables are related to each other by their common keys. The relationships between tables are determined by the product specification.

5.2.2.2.1 Feature definition. A feature is represented by a set of one or more primitives, a single row of attribute data in a feature table which uniquely identifies the feature, and zero or more rows of attribute data in other tables. A simple feature (e.g., a building) may consist of one or more primitives of a single type and a single row of attribute data. A complex feature (e.g., an airport) will be identified by one row in a complex feature table, but will include the additional information contained in other feature tables.

Features are grouped into feature classes. Each feature class is individually defined by a set of attributes (column definitions) and is uniquely named. The rows of features in a feature class collectively form the feature table for the feature class. Every feature class has one and only one feature table. The feature table is a special form of an attribute table because it directly references a feature. TABLE 5 expresses the basic structure of a feature table in VPF.

TABLE 5. Feature table structure.

Primary Key	Attributes
Either a primitive row identifier or feature definition table id (may be the table id).	Attributes as specified in the product specification, or join values for reference into other attribute tables.

5.2.2.2.2 Feature table joins. Simple features may be composed of one or more primitives of a single type, while complex features may be composed of one or more simple or complex features. A feature join designates which primitives belong to which features. Four types of feature joins represent all the possible relationships between features and primitives: one-to-one, many-to-one, one-to-many, and many-to-many. APPENDIX C provides a detailed discussion of these four types of join columns and feature join tables.

5.2.2.2.3 Feature class types. There are two types of feature classes in VPF: simple feature classes and complex feature classes. FIGURE 7 portrays the structural schema of these feature classes.

a. Simple feature classes. A simple feature class consists of a (logically) single primitive table and a single simple feature table. There are four subtypes of the simple feature class in VPF:

(1) Point feature classes (composed of entity or connected nodes)

(2) Line feature classes (composed of edges)

(3) Area feature classes (composed of faces)

(4) Text feature classes. A text feature class consists of a text primitive table and a text feature table. The text feature class is not a true feature class, but it is often useful to process text as if it were a feature. For instance, many maps contain text annotation that does not reference a specific geographic entity. The text "Himalaya Mountains" may not define any geometric primitive or feature, but merely provide associative information for the viewer. Using a text feature allows thematic queries on text just like other features. For instance, if a text feature has a height attribute, software can retrieve 'all text with HEIGHT > 0.5'.

b. Complex feature classes. A complex feature class consists of one or more simple feature classes, one or more complex feature classes, or both, and a single complex feature table, all within one coverage. For example, a complex watershed feature may be constructed from simple features, such as rivers, springs, and lakes.

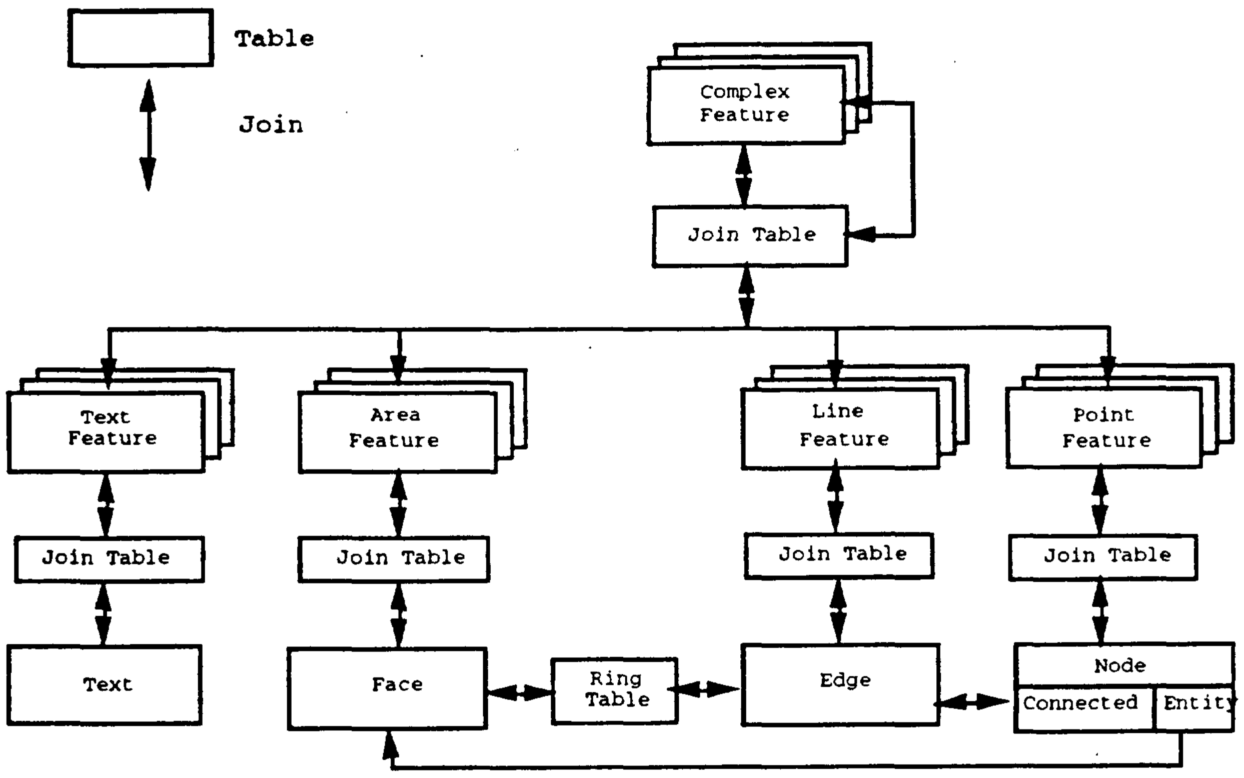


FIGURE 7. Feature class structural schema.

5.2.2.2.4 Constructing feature classes. A feature class consists of a set of tables that includes at least one primitive table and one feature table and optionally, join tables and related attribute tables. The rules for constructing feature classes are stored in the feature class schema table, which describes how each table relates to each other table in the feature class (TABLE 6).

TABLE 6. Feature class schema table.

Column Name	Description
ID	Required row id
FEATURE_CLASS	Name of the feature class
table1	The first table name in the relationship
table1_KEY	Column name of table 1 join key
table2	The second table name in the relationship
table2_KEY	Column name of table 2 join key

TABLE 7 shows a feature class schema table and an example of a simple feature class. Within the schema table, the feature class is named TRNLINE. The first table in the relation is called TRNLINE.LFT. The second table is named EDG, which is the standard label for the edge primitive. The key column, ID, in TRNLINE.LFT relates to the key column, ID, in EDG. TRNLINE.LFT has six attributes: F_CODE, BOT, LEN, OHB, TUC, and FROM_TO. F_CODE, BOT, LEN, OHB, and TUC are all feature attribute coding catalogue (FACC) codes for a feature. FROM_TO, on the other hand, describes the geometry of the feature (see section 5.3.3.1). The edge primitive contains the required columns for level 2 topology (see section 5.2.2.3.1). Appendix C provides additional information on feature classes and feature joins.

TABLE 7. Feature class schema table and simple feature class

Feature class schema table

ID	1	2
FEATURE_CLASS	TRNLINE	TRNLINE
TABLE1	TRNLINE.LFT	EDG
TABLE1_KEY	ID	ID
TABLE2	EDG	TRNLINE.LFT
TABLE2_KEY	ID	ID

Simple feature class

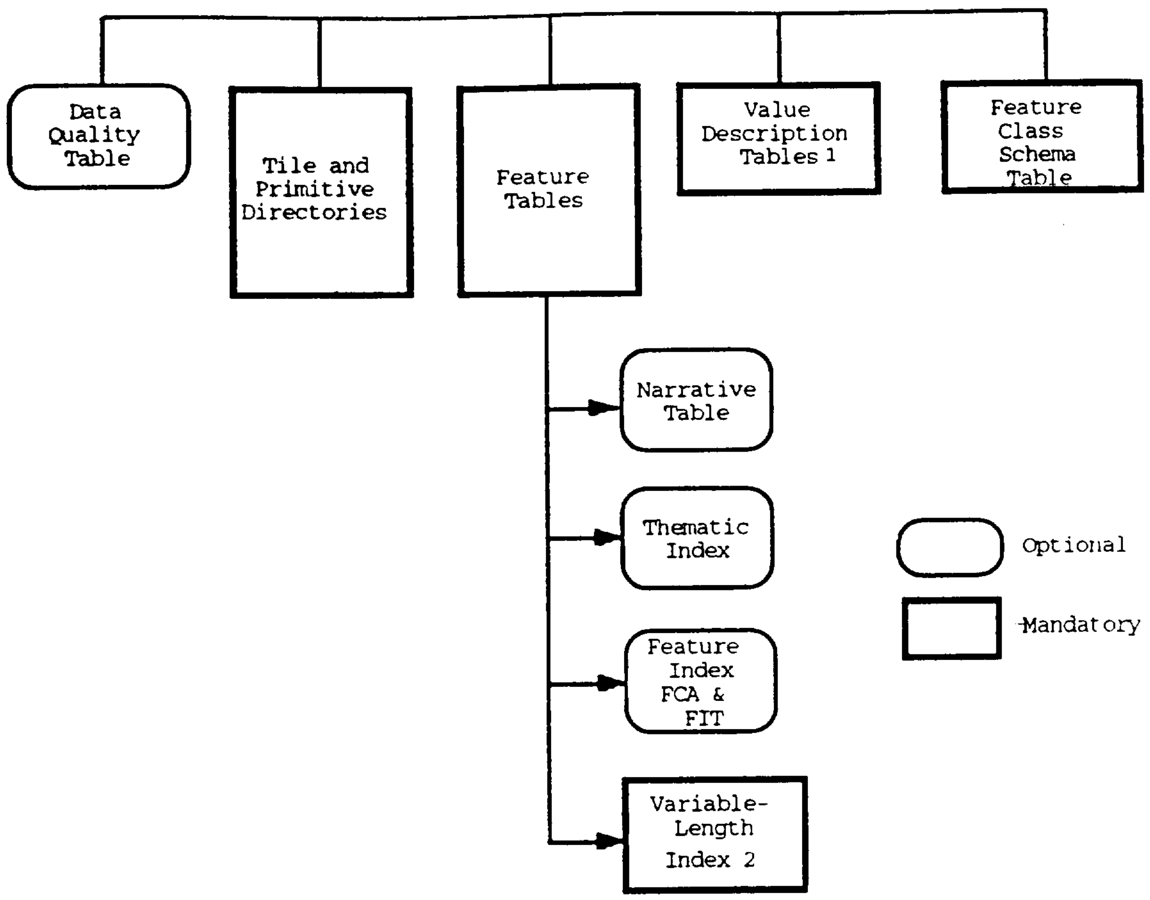
TRNLINE.LFT	
ID	1
F_CODE	AQ040
BOT	4
LEN	9
OHB	8
TUC	2
FROM_TO	1
ID	2
F_CODE	AQ040
BOT	0
LEN	5
OHB	4
TUC	3
FROM_TO	-1
ID	3
F_CODE	AQ040
BOT	4
LEN	7
OHB	2
TUC	4
FROM_TO	1

◄ ►

EDG	
ID	1
START_NODE	4
END_NODE	2
RIGHT_EDGE	9
LEFT_EDGE	2
COORDINATES	-97.706184,31.249201
	-97.706001,31.249952
	-97.706001,31.250172
ID	2
START_NODE	4
END_NODE	5
RIGHT_EDGE	5
LEFT_EDGE	1
COORDINATES	-97.706184,31.249201
	-97.702660,31.248232
ID	3
START_NODE	1
END_NODE	6
RIGHT_EDGE	7
LEFT_EDGE	8
COORDINATES	-97.734131,31.250172
	-97.734001,31.247892
	-97.733795,31.247061
	-97.733360,31.246422

5.2.2.3 Coverage. A coverage is composed of features whose primitives maintain topological relationships according to a level of topology (level 0, 1, 2, or 3) defined for the coverage. All of the file structures that make up a coverage are stored in a directory or subdirectories of that directory. A coverage is generally analogous to a photographic separate in conventional cartography.

At the coverage level (see FIGURE 8), there are three mandatory components: the primitive files or the subdirectories containing those primitives, the feature tables, and the feature class schema table. Value description tables must be used when implementing coded attributes. A variable length index file is mandatory whenever a variable length column is defined in a table. An MBR is required for each face and edge table. Spatial indexes are optional for each primitive table. A feature minimum bounding rectangle table may be included for each feature table. Maintaining a data quality table at the coverage level is optional. Feature index tables may be used to support quick retrieval of feature information for a selected primitive. Feature class attribute tables are required to support feature index tables. When tile directories exist, the primitive tables are placed in the tile directories. Tile directories are mandatory for a tiled coverage.



1. mandatory when coded attributes are used
2. mandatory when variable length column is defined in a table

FIGURE 8. Coverage contents.

5.2.2.3.1 VPF topology. There are four recognized levels of topology in VPF coverages, ranging from level 3, where all topological connections are explicitly present, to level 0, where no topological information is explicitly present. FIGURE 9 summarizes the characteristics of these levels and gives an example of each. Since text does not have any topological relationships, it is not listed in FIGURE 9. Text may be included with other primitives at any topological level, even though it does not have any topology.

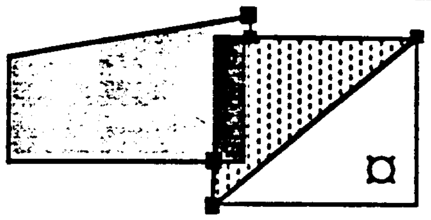
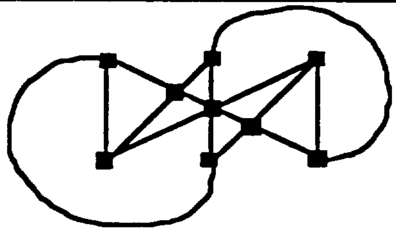
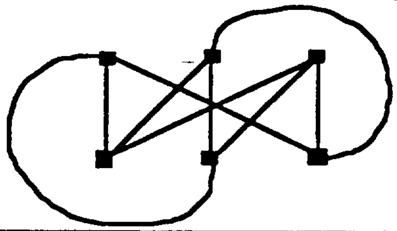
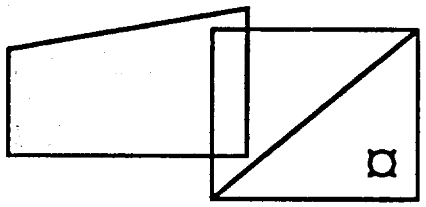
Level	**Name**	**Primitives**	**Description**	**Example**
3	Full topology	Connected nodes, entity nodes, edges, and faces	The surface is partitioned by a set of mutually exclusive and collectively exhaustive faces. Edges meet only at nodes.	
2	Planar graph	Entity nodes, connected nodes, and edges	A set of edges and nodes where, when projected onto a planar surface, the edges meet only at nodes.	
1	Non-planar graph	Entity nodes, connected nodes, and edges	A set of entity nodes and edges that may meet at nodes.	
0	Boundary representation (spaghetti)	Entity nodes and edges	A set of entity nodes and edges. Edges contain only coordinates, not start and end nodes.	

FIGURE 9. Levels of topology in VPF coverages.

The columns carried in the edge and node tables, which determine connectivity and adjacency for the topology, depend on the level of topology. For instance, the edge table in TABLE 7 does not contain the level 3 topology columns RIGHT_FACE and LEFT_FACE, because faces do not exist in level 2 topology. TABLE 8 shows the columns that are mandatory in each primitive table for the required level of topology. The characteristics of these columns are specified in the primitive definitions found in section 5.3.2.

TABLE 8. Columns required to define topology in VPF coverages.

Level	Primitive	Mandatory Columns
3	Face	RING_PTR
3	Ring Table	FACE_ID, START_EDGE
3	Edge	START_NODE, END_NODE, RIGHT_FACE, LEFT_FACE, RIGHT_EDGE, LEFT_EDGE
3	Entity Node	CONTAINING_FACE
3-1	Connected Node	FIRST_EDGE
2-1	Edge	START_NODE, END_NODE, RIGHT_EDGE, LEFT_EDGE
2-0	Entity Node	(none)
0	Edge	(none)

FIGURES 10, 11, and 12 use entity relationship (ER) diagrams to portray the primitives and their relationships for each level of topology.

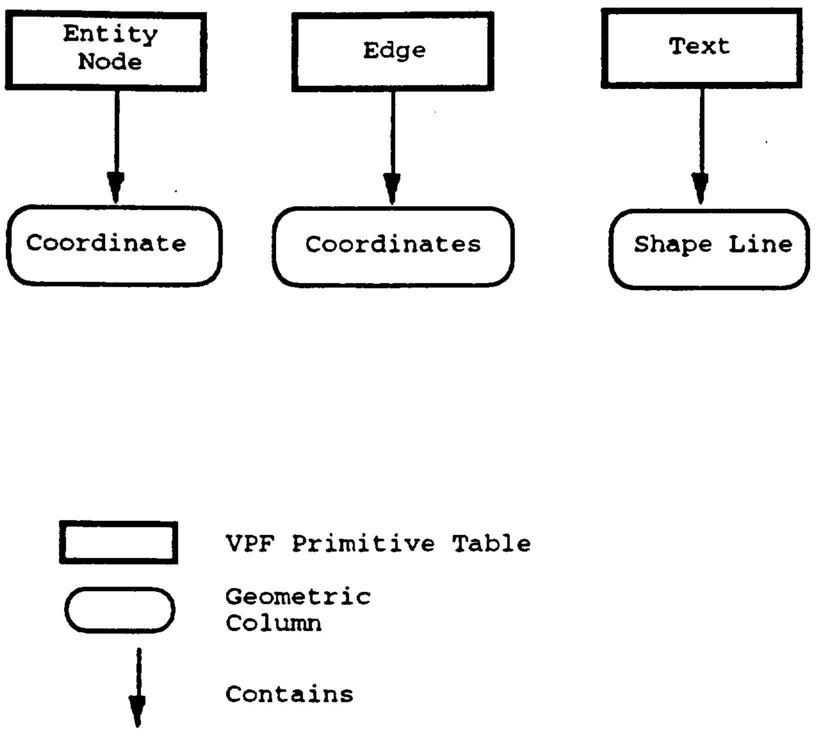


FIGURE 10. Level 0 topology.

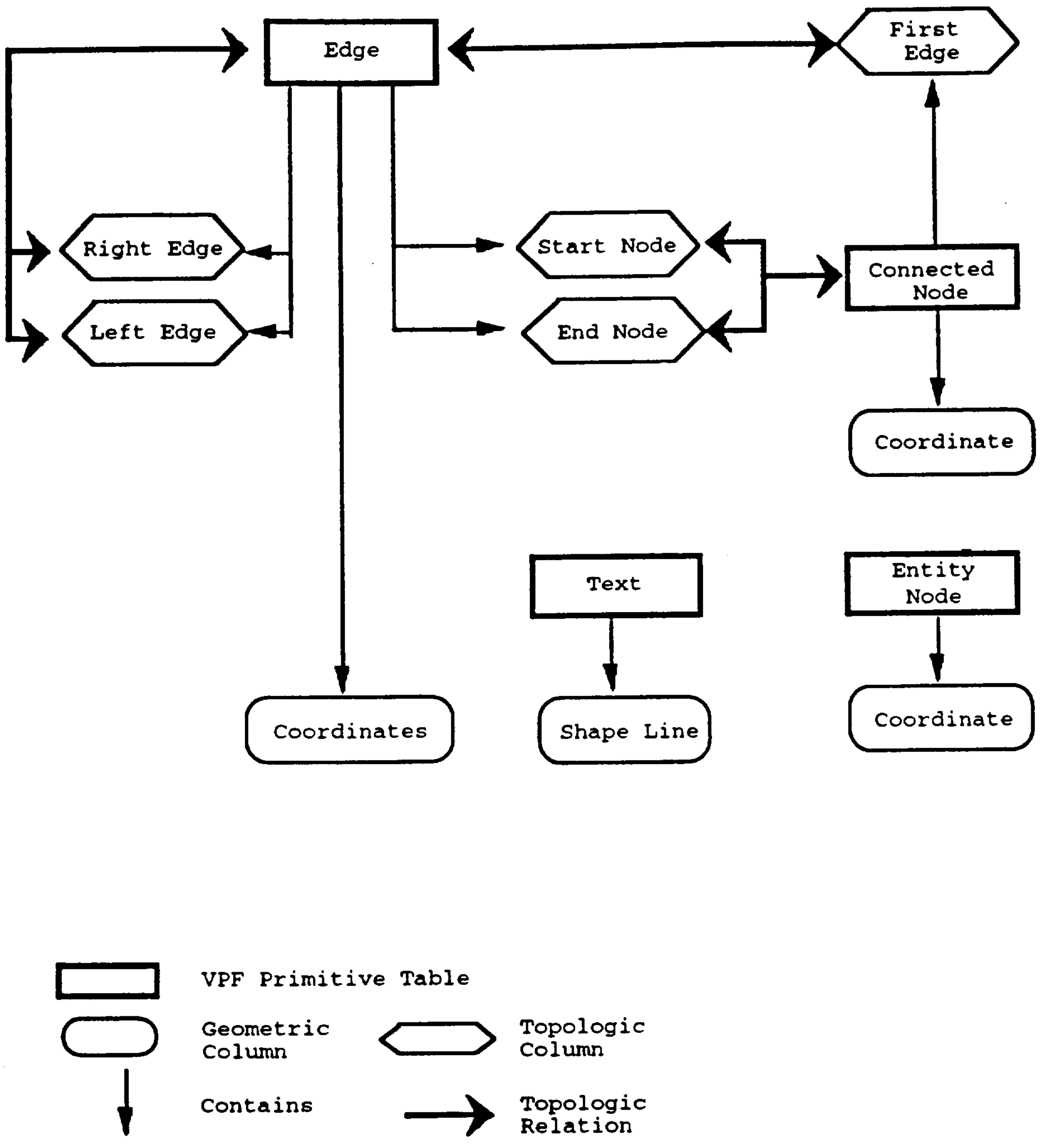


FIGURE 11. Level 1 and Level 2 topology.

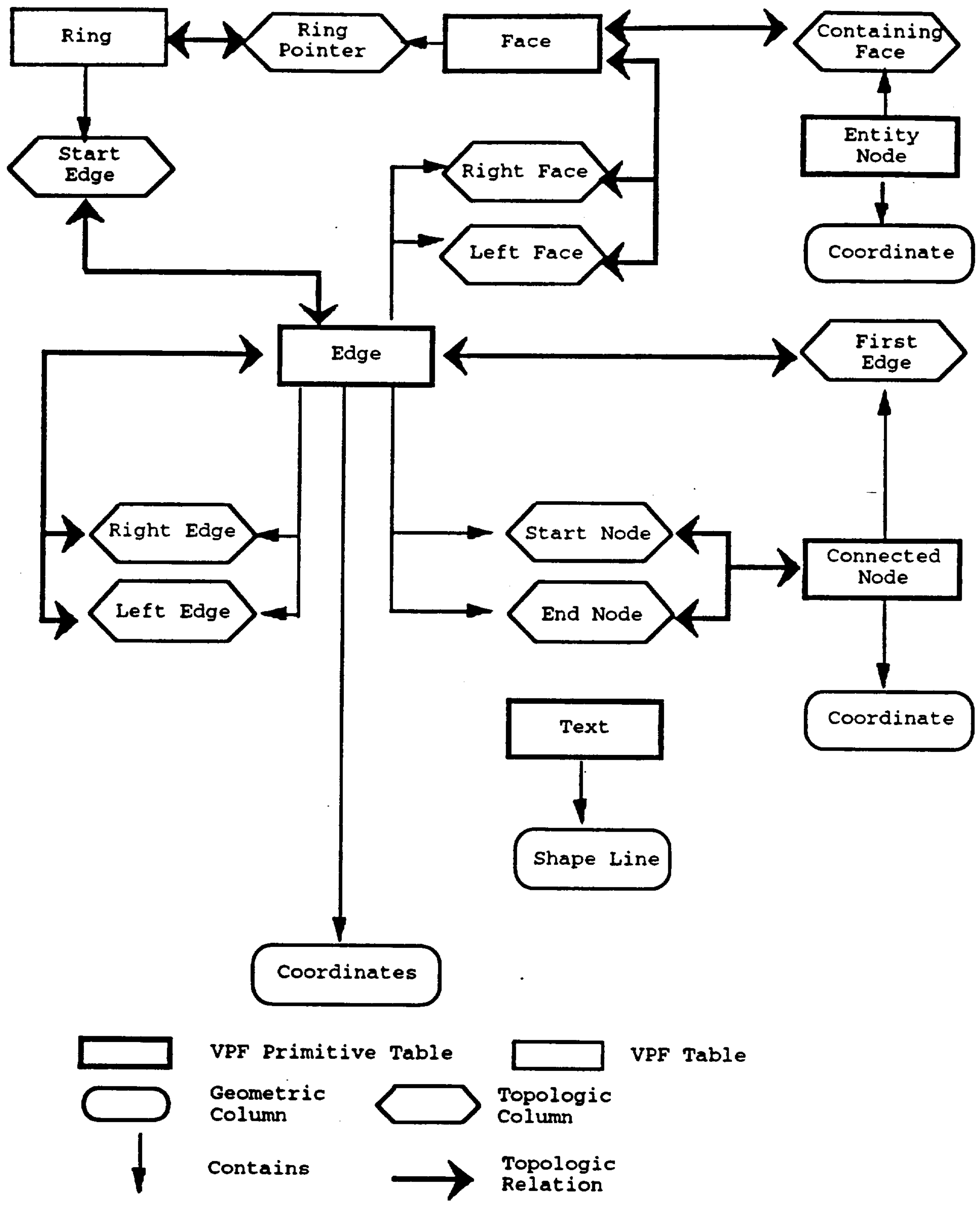


FIGURE 12. Level 3 topology.

5.2.2.3.2 Value description tables. A value description table (VDT) is provided to describe coded attributes. There are three types of attribute values: distinct values, integer value codes, and character value codes.

a. Integer value codes. In many cases, the values entered in an attribute column are only codes designed to facilitate data processing and transmission. Numerical codes and their corresponding descriptions are maintained in the integer VDT.

b. Character value codes. For alphanumeric codes, there is a character VDT similar to the integer VDT. For instance, feature attribute coding systems generally use a five-character-string feature coding scheme.

c. Distinct values. Distinct values are attribute values that can be directly interpreted. Measurements of length or elevation are examples of distinct values. The interpretation of distinct values does not require a value description table.

5.2.2.3.3 Tiled coverages. Tiling is geographically subdividing a coverage solely for the purpose of enhancing data management; a coverage subdivided in such a manner is then referred to as a tiled coverage. A tiled coverage contains the same attribute information as an untiled coverage. The logical interpretation of a tiled coverage is identical to that of an untiled one. Each tile will be a separate subdirectory under the coverage directory and contain separate primitive tables for those features contained within the tile. A tiled coverage will contain a single feature table for each feature class. Features in this table are joined with their corresponding primitives using a combination of tile ID and primitive ID. There should be no subdirectory carried in a coverage directory for any tile that is devoid of data in that coverage. However, the existence of face 1 justifies a tile subdirectory. Tiles do not contain feature attribute or schema tables. These tables belong to the coverage as a whole.

A tiled coverage is physically subdivided into tiles according to a tiling scheme. The tiling scheme (tile boundaries and size of tiles) and the handling of the features that lie on tile boundaries and text primitives that cross borders are all defined by a product specification. Each tile in a tiling scheme has a unique tile identifier. FIGURE 14 shows a tiling scheme that uses regular rectangular tiles. APPENDICES B (Winged-edge topology) and D (Tiling) contain more information on tiling and its' impact.

Primitive definition occurs wholly within a tile. The following paragraphs address the effect of tiling:

(1) Edges: When an edge is broken by a tile boundary a connected node is placed at the edge-tile intersection. The identical (geographic coordinate) connected node occurs in both tiles forming the boundary. All edges which lie along a tile boundary, will have cross-tile topology. The identical (geographic coordinates) edge occurs in both tiles forming the boundary.

(2) Faces: A face broken by a tile boundary has a new edge constructed and inserted at the boundary for each tile to close the face internal to the tile. These edges take part in cross-tile topology.

(3) Face 1: Face 1 (universe face) represents a special case for tile boundaries. In those cases where face 1 is the only face being broken, actual tile boundaries will not be stored. For example, where face 2 is broken by the tile boundary and the rest of the tile is defined by face 1, only the tile boundary edges necessary to close face 2 are stored (FIGURE 13).

(4) Connected Nodes: All connected nodes which lie on a tile boundary will have cross-tile components (tile_id and first_edge).

Two other situations to consider are that of a tile of a level 3 topology coverage which contain only point features or no features. In these cases, the tile contains either entity node primitives and face 1 or simply face 1, respectively. Level 3 topology requires inclusion of a face, ring, edge, connected node and face bounding rectangle table, and an edge variable lenght index (TABLE 8). The face, ring and face bounding rectangle tables will reference the universe face (face 1) only. The edge table must exist since it is referenced by the ring table. The connected node table must exist since it is referenced by the edge table. The existence of an edge table requires an edge variable lenght index and the existence of a face table requires a face bounding rectangle table. The edge and connected node table and the edge variable lenght index will contain no records. For this scenario the table appear as:

fac table		
id	dnarea.aft_id	ring_ptr
1	(NULL)	1

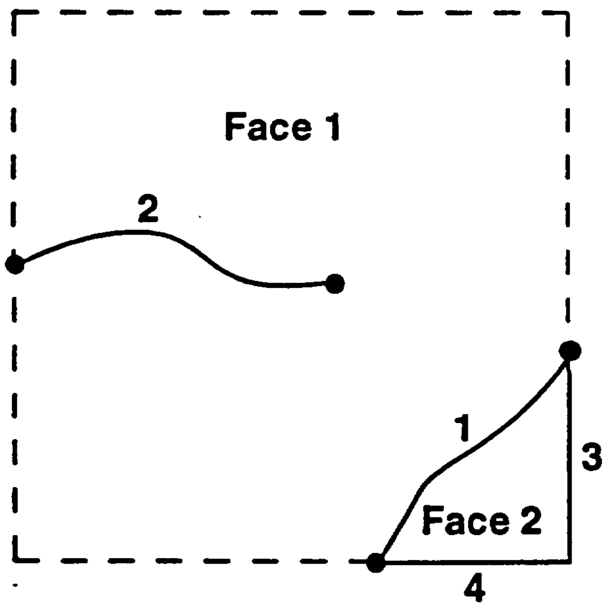
rng table		
id	fac_id	start_edge
1	1	(NULL)

fbr table				
id	x min	y min	x max	y max
1	(null)	(null)	(null)	(null)

edg table						
id	dnline.lft_id	sn	en	rf	lf	coordinates
(no records---header information only)						

cnd table				
id	dnpoint.pft_id	containing_face	first_edge	coordinate
(no records---header information only)				

end table			
id	dnpoint.pft_id	containing_face	coordinates
1	1	1	37.5,-76.5
2	5	1	39.0,-80.0



Note: Face 1 is the universe face. The tile's edge file will only store edges 1,2, 3 and 4. The dashed edges for the universe face are implied, but not stored.

FIGURE 13. Storage of tile boundaries.

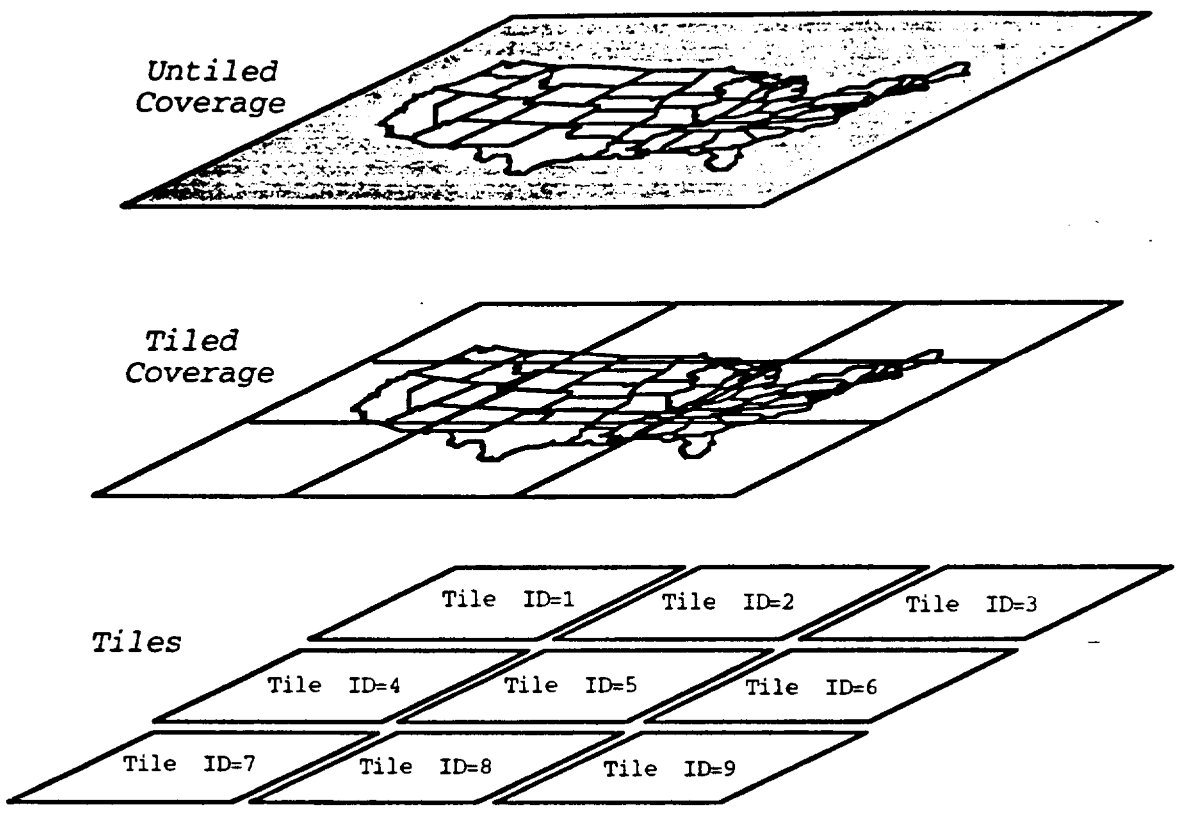


FIGURE 14. A tiling scheme.

5.2.2.3.4 Cross-tile keys. VPF provides a mechanism for maintaining geographic features in a logically continuous spatial database, whether or not a tiling scheme is present. Since the primitives in each tile of a tiled coverage are managed separately from those in other tiles, labels given to primitives are unique only within a tile. In order to support a logically continuous spatial database, a triplet id can be used instead of an integer key to reference primitives across multiple tiles. The triplet id augments the key of a primitive with the key of the tile in which the primitive falls. APPENDIX B contains a discussion that fully describes this concept.

a. For an edge primitive, the triplet id is used to maintain cross-tile topology. The Left Face, Right Face, Left Edge, and Right Edge columns are defined as triplet ids to support tiled coverages. The triplet id contains a reference to the internal topology within the current tile; the two other components reference the external tile directory and the primitive within that tile. For example, for a face divided by a tile boundary, the external id portion of the Left Face field in FIGURE 15 would include the continuing face in the other tile. This inclusion of internal and external tile references allows software to detect tile borders and continue operations across boundaries or to operate only within the current tile. If a coverage is untiled, the Left Face, Right Face, Left Edge, and Right Edge columns may be defined as integer columns; otherwise the external tile id and primitive id sub-fields of the triplet id will not exist (see 5.4.6.)

b. Cross-tile topology only occurs between tiles within a library. Cross-tile components will only be populated for edges intersecting tile boundaries within a library. Edges on tile boundaries which coincide with library boundaries will not have cross-tile components populated.

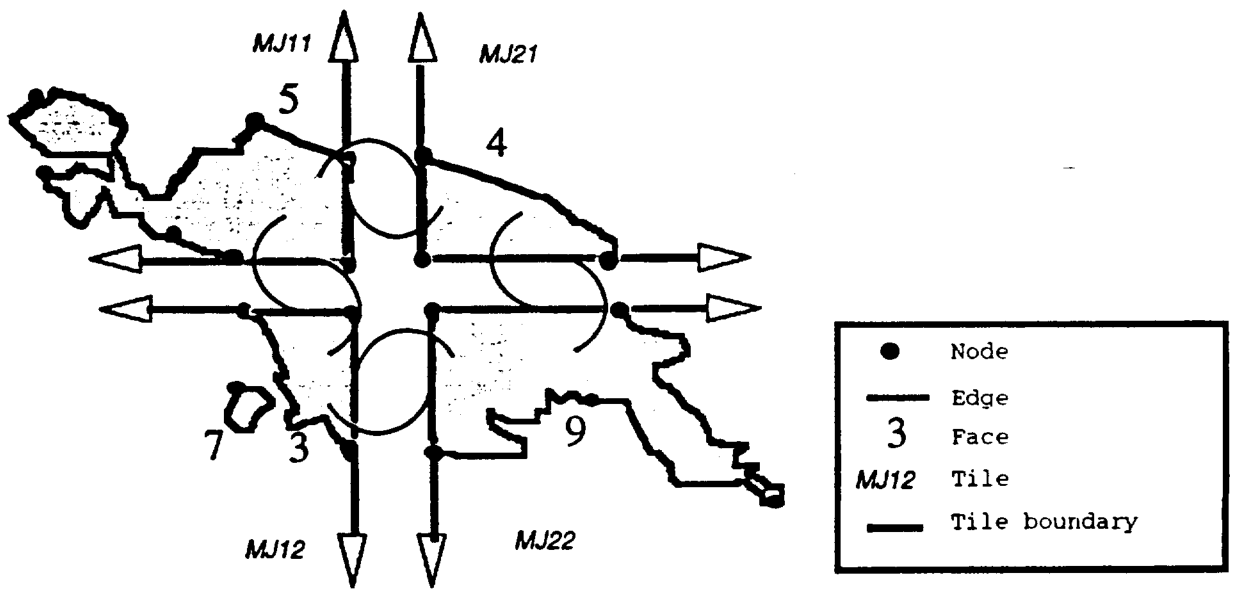


FIGURE 15. Face cross-tile matching.

5.2.2.4 Library. A library is a collection of coverages that share a single coordinate system and scale, have a common thematic definition, and are contained within a specified spatial extent. If any of the coverages composing the library are tiled, then all other coverages must either use the same tiling scheme, or be untiled. The contents and organization of the libraries are determined by a product specification. All of the tables and coverages making up the library are contained within a single master directory (FIGURE 16).

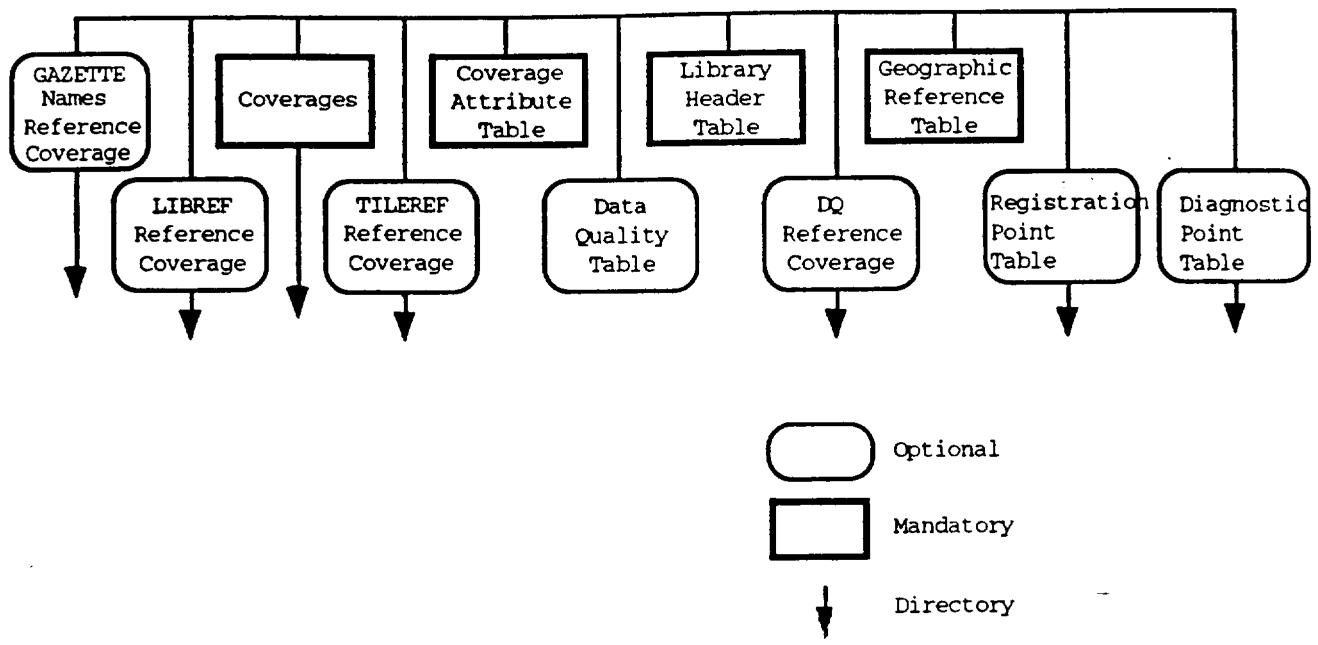


FIGURE 16. Library directory.

5.2.2.4.1 Tile reference coverage. A tile reference coverage is mandatory if a library contains tiled coverages. The spatial extent of the library and its tiling scheme are represented in the tile reference coverage. A library can not contain partial tiles. This reference coverage contains a set of faces and area features identifying the tiles that the library uses to subdivide the region of interest. The universe face always has a face id that equals one. The inner ring for face 1 in TILEREF defines the library's spatial extent. For irregularly shaped libraries, this will be a smaller total area than the bounding rectangle defined in the LAT. The tile reference coverage is a standard untiled coverage with level 3 topology.

5.2.2.4.2 Library attributes. General information about a library is stored in the library header table. There is one primary attribute row per library, and zero or more other attribute rows. Libraries are also the level at which coverage attribute tables reside. The coverage attribute table identifies the coverages found in a library.

5.2.2.4.3 Library coordinate system. The coordinate system of a library is defined by a geographic reference table. This table defines the basic coordinate system for the library. Extensions to the basic coordinate system may be provided by a product specification.

An example of a geographic reference table would document the projection used, its base parameters, and the values used to define the size of the Earth. This information (values for the semi-major and the semi-minor axis of an ellipsoid, other projection datum information, the false origin of a projection, and so forth) is necessary to understand a coordinate system in a VPF library.

5.2.2.4.4 Library reference coverage. When tiles exist in the library, a library reference coverage must exist. This coverage is spatially registered to the tile reference coverage to provide a preliminary view of the data contained within the library to use for such functions as "zoom out." The contents of this coverage will be a generalized map of the coverage considered to be most significant to the library. For example, if a library contains the rivers, transportation, and political boundaries of the Australian continent, a generalized map of the political boundaries might be considered appropriate for the library reference coverage.

5.2.2.4.5 Data quality reference coverage. It is possible to include a data quality coverage at the library level. This coverage is spatially registered to the tile reference coverage. Its purpose is to record data quality information that pertains to the entire library. Appendix E contains more detailed information about the contents of this coverage.

5.2.2.4.6 Names reference coverage. The names reference coverage provides the user with a way to locate a place in a library by using a place name. This is a special type of thematic query. The most common use of the names reference coverage is to enter a query string (for instance "London"), have the software locate all the places that are "London," and display their geographic locations and names on the display device. The name feature class contains a point feature table and an entity node primitive table.

5.2.2.5 Database. A database is a collection of related libraries and additional tables. The library attribute table acts as a table of contents for the database. Database transmittal information is contained in a database header table. Database level data quality information can be maintained in the data quality table. Appendix E contains more detailed information about the content of this table. FIGURE 17 illustrates the arrangement of database tables and coverages.

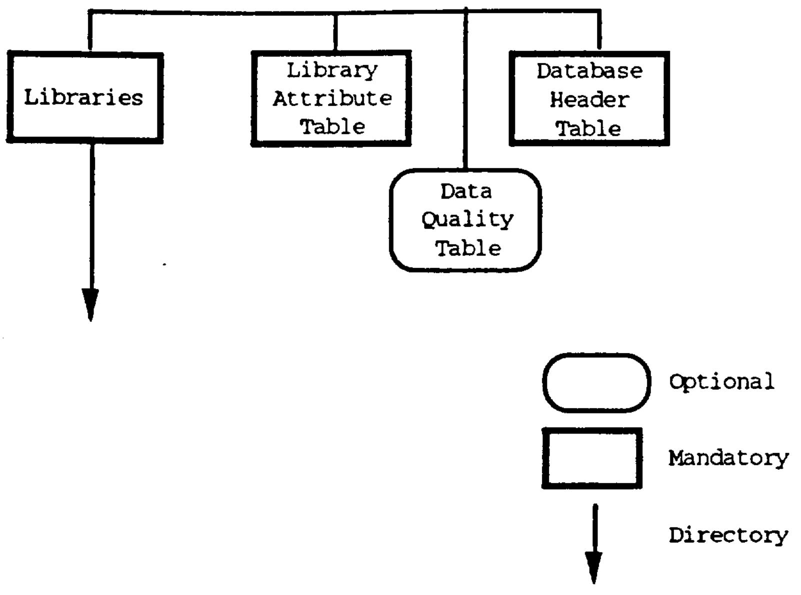


FIGURE 17. Database directory.

5.2.3 Data quality. VPF allows for the storage of data quality information to permit the evaluation of the data for particular applications. Although the exact form of the data quality information supplied for a database is set by a product specification, VPF supports incorporation of data quality information at each structural level in the database. Data quality information may be stored at any VPF level. When it exists at a given level, it applies to all data at or below that level. However, when data quality information exists at multiple levels, the information stored at lower levels always takes precedence over that at the higher levels.

5.2.3.1 Types of data quality information. A VPF database may contain seven types of data quality information: source, positional accuracy, attribute accuracy, date status, logical consistency, feature completeness, and attribute completeness. Definitions of these quality types are provided in appendix E. The extent of data quality information contained in a product and the types of data quality to be included are determined by the product specification.

5.2.3.2 Data quality encoding. Data quality information can be represented as an attribute or as a coverage. In the case of attributes, data quality information may be added to an existing VPF table, stored in a separate table, or stored in the data quality table discussed in section 5.3.7. APPENDIX E describes data quality encoding in more detail.

5.3 Implementation. The following paragraphs describe the implementation requirements of the VPF data structures. Discussion covers the primitive, feature class, coverage, library, and database levels. A description of a data quality table and the narrative table is also provided.

5.3.1 General implementation information. In order to fully explain the content of each data structure, each table is given a text description, definition table, and an example.

5.3.1.1 Table definitions. These column descriptions define the contents of each table. Each description example contains five entries: column name, description, column type, key type, and whether the column is optional or mandatory (Op/Man; see TABLE 12). The asterisk (*) in the column name item is a substitute for an associated feature or primitive table name that is provided by the product specification. "Null" in example tables refers to a valid VPF null for that column type. Example tables may not reflect the VPF requirement for consecutive row IDs. TABLE 9 is an example of the table definition style.

TABLE 9. Table definition example.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
*.PFT_ID	Feature id	I	N	OF
CONTAINING_FACE	Face containing the entity node	I	N	M3
FIRST_EDGE	(Null)	X	N	O
COORDINATE	Coordinates	C	N	M

Schema descriptions identify the following columns.

a. Column type. The column type column in the definition table expresses the type of data the column must contain. The encapsulation of these types is discussed in additional detail in section 5.4. For the purposes of this section, TABLE 10 identifies field types. When the number of elements is not specified as part of a column type definition described in this document for any VPF table header, it is assumed to be 1.

b. Key type. VPF provides three key types. They are primary keys, unique keys, and non-unique keys. Columns identified as non-unique in this document may be changed to unique by a product specification. TABLE 11 lists the key types and the codes used in table definitions. Any primary key may be referred to as a foreign key in another table.

TABLE 10. Column types.

Column Type	Description
T,n	Fixed-length text
T,*	Variable-length text
L,n	Level 1 (Latin 1 - ISO 8859) Fixed-length text
L*	Level 1 (Latin 1 - ISO 8859) Variable-length text
N,n	Level 2 (Full Latin - ISO 6937) Fixed-length text
N*	Level 2 (Full Latin - ISO 6937 Variable-length text
M,n	Level 3 (Multilingual - ISO 10646) Fixed-length text
M*	Level 3 (Multinlingual - ISO 10646) Variable-length text
F	Short floating point
R	Long floating point
S	Short integer
I	Long integer
C,n	2-coordinate array short floating point
C,*	2-coordinate string short floating point
B,n	2-coordinate array long floating point
B,*	2-coordinate string long floating point
Z,n	3-coordinate array short floating point
Z,*	3-coordinate string short floating point
Y,n	3-coordinate array long floating point
Y,*	3-coordinate string long floating point
D	Date and time
X	Null field
K	Triplet id

Note: The asterisk (*) indicates variable-length string. n indicates a fixed-length array; n is defined by the product specification. The product specification can change columns of type * to n. Type characters are case sensitive when used in table definitions.

TABLE 11. Key types.

Key	Description
P	Primary key
U	Unique key
N	Non-unique key

c. Optional/mandatory. The optional/mandatory column indicates whether the column is optional or mandatory for a VPF table. For each column, there are several mandatory conditions, as shown in TABLE 12. The code OF is used on a primitive table when direct pointers to the feature table are desired to improve performance.

TABLE 12. Optional/mandatory conditions.

Code	Description
O	Optional
OF	Optional feature pointer
M	Mandatory
M<n>	Mandatory at level n topology (0-3)
MT	Mandatory if tiles exist

5.3.1.2 Reserved table names and extensions. Each VPF table name consists of a reserved name or suffix extension. TABLE 13 lists the tables whose names cannot be modified or changed.

There are a few reserved directory names at the library and database levels. These names are listed in TABLE 14.

In a coverage directory, there are many feature class tables that have reserved suffixes. The product specification may define any eight-character prefix, following the naming conventions detailed in section 5.4.5. TABLE 15 lists the table suffixes.

TABLE 13. Reserved file names.

File Name	Description
cat	Coverage Attribute Table
cnd	Connected Node Primitive
csi	Connected Node Spatial Index
dht	Database Header Table
dqt	Data Quality Table
ebr	Edge Bounding Rectangle
edg	Edge Primitive
end	Entity Node Primitive
esi	Edge Spatial Index
fac	Face Primitive
fbr	Face Bounding Rectangle
fca	Feature Class Attribute Table
fcs	Feature Class Schema Table
fsi	Face Spatial Index
grt	Geographic Reference Table
lat	Library Attribute Table
lht	Library Header Table
nsi	Entity Node Spatial Index
rng	Ring Table
txt	Text Primitive
tsi	Text Spatial Index
char.vdt	Character Value Description Table
int.vdt	Integer Value Description Table

TABLE 14. Reserved directory names.

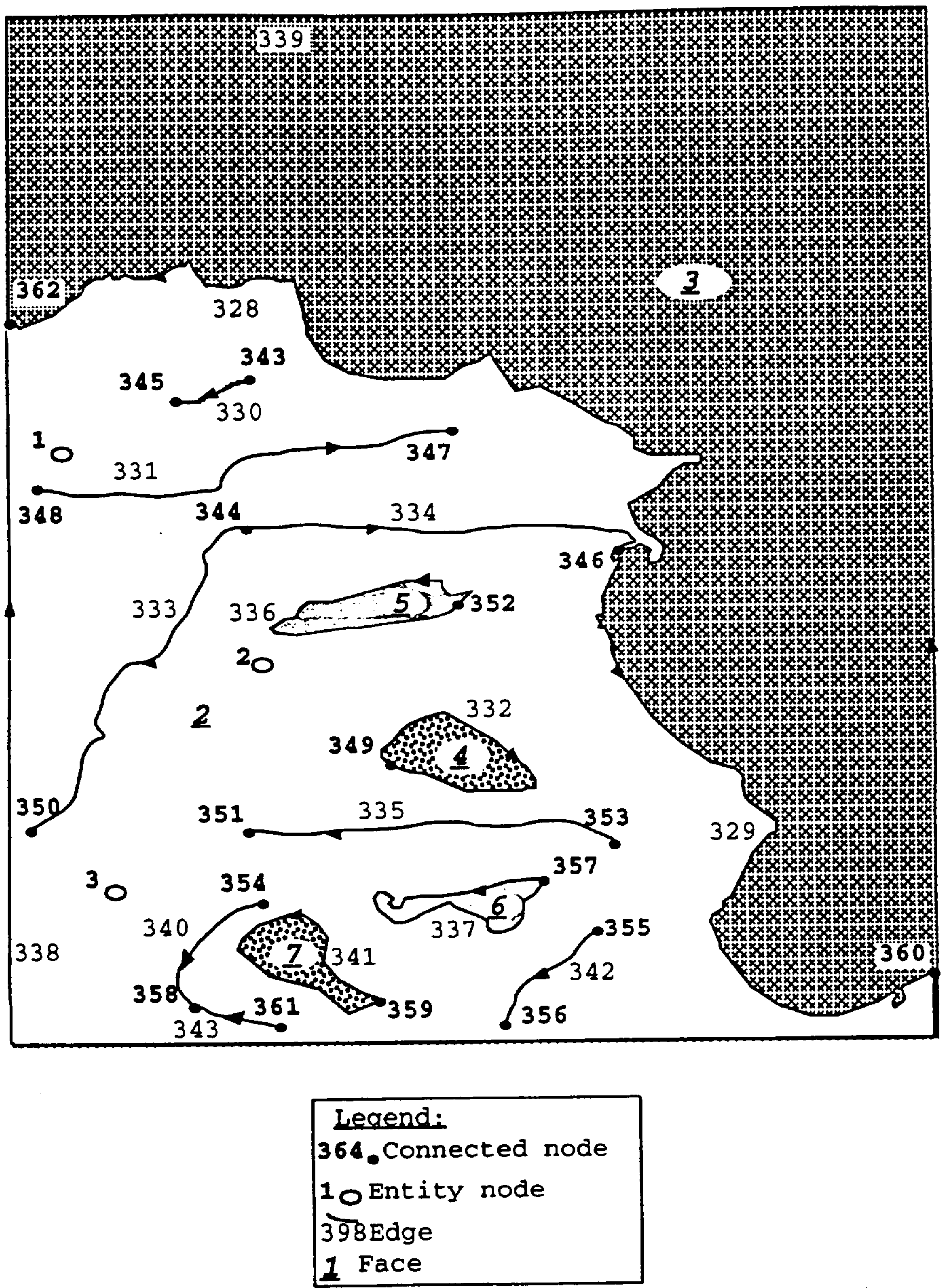
Directory Name	Description
libref	Library reference coverage
dq	Data quality coverage
tileref	Tile reference coverage
gazette	Names reference coverage

TABLE 15. Reserved table name extensions.

File Name Suffix	Description
.abr	Area Bounding Rectangle Table
.aft	Area Feature Table
.ajt	Area Join Table
.ati	Area Thematic Index
.cbr	Complex Bounding Rectangle Table
.cft	Complex Feature Table
.cjt	Complex Join Table
.cti	Complex Thematic Index
.doc	Narrative Table
.dpt	Diagnostic Point Table
.fit	Feature Index Table
.fti	Feature Index Table Thematic Index
.jti	Join Thematic Index
.lbr	Line Bounding Rectangle Table
.lft	Line Feature Table
.ljt	Line Join Table
.lti	Line Thematic Index
.pbr	Point Bounding Rectangle Table
.pft	Point Feature Table
.pjt	Point Join Table
.pti	Point Thematic Index
.rat	Related Attribute Table
.rpt	Registration Point Table
.tft	Text Feature Table
.tti	Text Thematic Index

Any table that contains variable-length records must have a variable-length index associated with it. The index file shall have the same file name as the table, except that the last character will end with "X." For example, a variable-length record road line table, ROAD.LFT, would have a variable-length index ROAD.LFX. The one exception to this convention is for the FCS, whose variable-length index shall be named FCZ.

5.3.2 Primitives. As discussed in section 5.2.2.1, there are three types of geometric primitives in VPF: nodes, edges, and faces. There are two classes of nodes, the entity node and the connected node. In addition, text is used as a cartographic primitive. These four primitives, with the addition of feature tables, allow the modeling of geographic phenomena requiring vector geometry. FIGURE 18 illustrates the various types of primitives. Columns can only be added to primitive tables to handle SOURCE, POSITIONAL ACCURACY, UP-TO-DATENESS, SECURITY, and RELEASABILITY.



(Note: This FIGURE represents a partial database. Therefore only a subset of primitives are shown. A complete set of primitives would have sequentially numbered IDs beginning with 1.)

FIGURE 18. Node, edge, and face primitives.

5.3.2.1 Node primitives. Two types of node primitives are implemented: entity nodes, which are free floating, and connected nodes, which occur only at edge ends. Both represent zero-dimensional locations.

a. Entity node primitive. The entity node primitive is composed of three columns: a primary key, a foreign (to the face table) key, and the node coordinates. The FIRST_EDGE null column is included to maintain compatibility with the connected node primitive so that the formats for both classes of node primitive conceptually remain the same. The CONTAINING_FACE column is only required for level 3 topology to maintain a topological relationship to the face that contains the node. TABLE 16 defines the meaning of the entity node primitive. TABLE 17 illustrates an entity node table; the entity nodes described are those in FIGURE 18.

TABLE 16. Entity node definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
*.PFT_ID	Feature id	I	N	OF
CONTAINING_FACE	Face containing the entity node	I	N	M3
FIRST_EDGE	(Null)	X	N	O
COORDINATE	Coordinates	C/Z/B/Y	N	M

Note: The asterisk (*) indicates a placeholder for the point feature class name.

TABLE 17. Entity node records.

Column Name	Contents
ID	1
DNPOINT.PFT_ID	936
CONTAINING_FACE	2
FIRST_EDGE	Null
COORDINATE	10.56 37.91
ID	2
DNPOINT.PFT_ID	937
CONTAINING_FACE	2
FIRST_EDGE	Null
COORDINATE	10.36 37.72
ID	3
DNPOINT.PFT_ID	953
CONTAINING_FACE	2
FIRST_EDGE	Null
COORDINATE	10.15 37.86

b. Connected node primitive. The connected node primitive is composed of three columns: a primary key, a foreign key (to the edge table), and the node coordinates. The CONTAINING_FACE null column is included to maintain compatibility with the entity node primitive. The FIRST_EDGE column is a foreign key required for level 1 and higher topology levels to maintain a topological relationship to the edges that include the node. The complete set of edges around a connected node may be assembled by following the topology of the connected node until the first edge reappears. Refer to APPENDIX B for more discussion of this algorithm. A connected node table is required for any coverage of topology level 1-3 containing an edge table. If the optional feature pointer is used, connected nodes that are not features will carry a null in that field. TABLE 18 defines the connected node primitive. TABLE 19 illustrates a connected node table with data for four of the connected nodes shown in FIGURE 18.

TABLE 18. Connected node definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
*.PFT_ID	Feature id	I	N	OF
CONTAINING_FACE	(Null)	X	N	O
FIRST_EDGE	Edge id	K/I	N	M1-3
COORDINATE	Coordinates	C/Z/B/Y	N	M

Note: The asterisk (*) indicates a placeholder for the point feature class name.

TABLE 19. Connected node records.

Column Name	Contents
ID	343
DN.PFT_ID	42
CONTAINING_FACE	Null
FIRST_EDGE	330
COORDINATE	10.63 37.72
ID	344
DN.PFT_ID	Null
CONTAINING_FACE	Null
FIRST_EDGE	333
COORDINATE	10.49 37.73
ID	345
DN.PFT_ID	Null
CONTAINING_FACE	Null
FIRST_EDGE	330
COORDINATE	10.61 37.80

5.3.2.2 Edge primitive. The edge primitive table includes up to eight mandatory columns, depending on the level of topology. The mandatory ID column contains the row id and is the primary key. The start_node and end_node columns are foreign keys to the connected node primitive and are mandatory for levels 1-3. The right_face and left_face columns are foreign keys to the face table and are mandatory for level 3 topology. The right_edge and left_edge columns are foreign keys to the edge table and are mandatory for levels 1-3. The coordinates column is mandatory. For simplicity in drawing edges, the coordinate string includes the node coordinates at each end, regardless of the existence of a connected node primitive. Thus, the minimum length of the coordinate string is two pairs. APPENDIX B describes winged-edge topology in more detail.

a. Node information. The foreign keys to the connected node table are required for level 1 and higher topology levels to maintain a topological relationship to the node connected to the edge. The start node indicates the beginning of the edge, and the relationship of the start node to the end node defines the edge orientation.

b. Edge information. Two foreign keys, right and left edge, are required for level 1, 2, and 3 topology, establishing connectivity between each edge and its neighboring edges in the coverage network. The right and left edges establish winged-edge topology for both line networks and faces. If all the edges incident at a node are sorted according to the bearing each edge radiates from that node, the right edge of a particular edge is the first edge encountered, counterclockwise in the sort order, around the end node of that particular edge. Similarly, the left edge is the first edge encountered around the start node.

c. Face information. When faces are present, the right and left face columns are added to the edge primitive table. Depending on the edge direction, the face columns are assigned. When a face is split by a tile boundary, the internal tile boundary is used to close the face on each tile. The tile id and external face id are also maintained in the triplet id.

TABLE 20, defines the edge table. TABLE 21 illustrates an edge table created from data shown in FIGURE 18.

TABLE 20. Edge table definition.

Column Name	Description	Field Type	Key Type	Op/Man
ID	Row id	I	P	M
**.LFT_ID	Feature id	I	N	OF
START_NODE	Start node id	I	N	M1-M3
END_NODE	End node id	I	N	M1-M3
RIGHT_FACE	Right face id	K/I	N	M3
LEFT_FACE	Left face id	K/I	N	M3
RIGHT_EDGE	Right edge id from end node	K/I	N	M1-M3
LEFT_EDGE	Left edge id from start node	K/I	N	M1-M3
COORDINATES	Coordinates	C/Z/B/Y,*	N	M

Note: The (**) indicates a placeholder for the line feature class name.

TABLE 21. Edge record example.

ID	LINE	SN	EN	RF	LF	REd	LEd	Coordinates Start Node...End Node
328	24500	346	362	3	2	339	334	10.46 37.38...10.68 37.00
329	24502	346	360	2	3	338	328	10.46 37.38...10.06 37.09
330	24505	343	345	2	2	330	330	10.63 37.72...10.61 37.80
331	24524	348	347	2	2	331	331	10.53 37.93...10.58 37.53
332	24534	349	349	4	2	332	332	10.26 37.36...10.26 37.36
333	24569	344	350	2	2	333	334	10.49 36.73...10.20 36.94
334	24573	344	346	2	2	329	333	10.49 36.73...10.46 37.38
335	24575	353	351	2	2	335	335	10.18 36.38...10.20 36.73
336	24581	352	352	2	5	336	336	10.20 36.81...10.20 36.81
337	24585	357	357	2	6	337	337	10.15 36.46...10.15 36.46
338	Null	360	362	2	1	328	339	10.06 37.09...10.68 37.00
339	Null	360	362	1	3	338	329	10.06 37.09...10.68 37.00
340	24601	354	358	2	2	343	340	10.13 36.72...10.04 36.78
341	24603	359	359	2	7	341	341	10.04 36.61...10.04 36.61
342	24612	355	356	2	2	342	342	10.10 36.40...10.02 36.50
343	24626	361	358	2	2	340	343	10.02 36.71...10.04 36.78

Note: LINE = **.LFT_ID, SN = START_NODE, EN = END_NODE, RF = RIGHT_FACE, LF = LEFT_FACE, REd = RIGHT_EDGE, LEd = LEFT_EDGE. Only start and end node coordinates are shown, although all coordinates would actually be present in this variable-length column.

5.3.2.3 Face primitive. Faces are defined as planar regions enclosed by an edge or a set of edges. All faces are defined by one or more rings, which are connected networks of edges that compose the face border. Each ring starts with a reference to a particular edge, and is defined by traveling in a consistent direction. Then the left and right edge columns on the edge primitive are traversed, always keeping the face being defined on one side, until the ring returns to its starting edge. All faces

must have one and only one outer ring, which bounds the exterior. A face may require inner rings to represent areas belonging to other faces that it encloses totally. Inner rings and outer rings are disjointed. There is no upper limit on the number of inner rings. A ring table (see below) is defined to handle these disjointed rings. A ring within a ring contained within a face (e.g., a lake within an island which is contained within a larger lake) has no direct topologic relation to the outer face (the larger lake). Face primitives are implemented as follows.

a. Face table. The face table contains two columns, where the primary key id identifies the face and the RING_PTR column points to the outer ring in the ring table. Face id 1 is always reserved for the universe face in a face table; it will never correspond to a feature in the feature table. The universe face contains a point at infinity. The outer ring of the universe face is a topological artifact which does not have a geometric representation. The outer ring cannot be displayed. The common boundary between the universe face and all other faces constitutes the inner ring or rings of the universe face. Inner rings of the universe face behave the same as the rings of other faces. TABLE 22 defines the face table. TABLE 23 depicts an example of the face table for three faces from FIGURE 18.

TABLE 22. Face table definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M3
*.AFT_ID	Feature id	I	N	OF
RING_PTR	Ring id	I	N	M3

Note: The asterisk (*) indicates a placeholder for the area feature class name.

TABLE 23. Face record example.

Column Name	Contents
ID	1
DNAREA.AFT_ID	Null
RING_PTR	1
ID	2
DNAREA.AFT_ID	4571
RING_PTR	2
ID	3
DNAREA.AFT_ID	4572
RING_PTR	3

b. Ring table. The ring table contains one reference to the edge table for each ring of a face. The first row in the ring table for each face refers to the outer ring of that face. Because the outer ring for face 1 (universe face) is a topological artifact, its start edge will be null. Outer rings are traversed in clockwise direction. Each inner ring has one reference to a first edge on that ring. The ring table maintains an order relationship for its rows. The first record of a new face id will always be defined as the outer ring. Any repeating records with an identical face value will define inner rings. TABLE 24 defines ring table structure, and TABLE 25 depicts three rings from FIGURE 18 and follows the face example in TABLE 23.

TABLE 24. Ring table definition.

Column Name	Description	Field Type	Key Type	Op/Man
ID	Row id	I	P	M3
FACE_ID	Face id	I	N	M3
START_EDGE	Edge id	I	N	M3

TABLE 25. Ring record example.

Column Name	Contents
ID	1
FACE_ID	1
START_EDGE	Null
ID	2
FACE_ID	2
START_EDGE	338
ID	3
FACE_ID	3
START_EDGE	329

5.3.2.4 Text primitive. Text is implemented to allow the representation of names associated with vague or ill-defined regions, such as the Appalachian Mountains. The text primitive is normally composed of three items: a primary key, the text string, and a coordinate string defining a shape line. Optional attributes may also be associated with the primitive, such as text color or font height.

The shape line must contain at least one coordinate pair. If the shape line contains only one coordinate pair, the coordinate pair is considered to represent the lower left coordinate, and the default orientation for the shape line will be assumed (minimum readable text and parallel to X axis.) In order to specify orientation, two coordinate pairs must be entered. The second coordinate pair defines the lower right of the string. Some fonts have ascenders and descenders that extend above or below the shape line. Third and subsequent coordinate pairs define control points in a shape line. The control points of a shape line define a continuous function. Characters in a text string are individually oriented along the shape line.

Table 26 defines the text primitive table structure. TABLE 27 is a hypothetical example of a text primitive table.

TABLE 26. Text primitive structure table.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
**.TFT_ID	Feature id	I	N	OF
STRING	Text string	T/L/M/N,*	N	M
SHAPE_LINE	Coordinates	C/Z/B/Y,*	N	M

Note: The (**) indicates a placeholder for the text feature class name.

TABLE 27. Text primitive record example.

Column Name	Contents
ID	1
DNTEXT.TFT_ID	529
STRING	Fiume Salso
SHAPE_LINE	14.41,37.74 14.45,37.73 14.52,37.69 14.60,37.69
ID	2
DNTEXT.TFT_ID	530
STRING	Simeto
SHAPE_LINE	14.80,37.71 14.81,37.68 14.80,37.66 14.81,37.65
ID	3
DNTEXT.TFT_ID	531
STRING	Belice
SHAPE_LINE	12.91,37.69 12.93,37.71 12.95,37.73
ID	4
DNTEXT.TFT_ID	532
STRING	Dittaino
SHAPE_LINE	14.51,37.56 14.54,37.55 14.58,37.56 14.62,37.57

5.3.2.5 Minimum bounding rectangle table. A minimum bounding rectangle record is required for each record in an edge or face primitive table. Since the outer ring of the universe face is a topological artifact which does not have a geometric representation, the FBR record for face 1 should contain nulls for XMIN, YMIN, XMAX and YMAX. The definition found in TABLE 28 is used for both the face and edge minimum bounding rectangle tables. TABLE 29 is an example of the face minimum bounding rectangle table used for FIGURE 18. The MBR table definition is applicable to both edge and face primitives.

TABLE 28. Minimum bounding rectangle definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
XMIN	Minimum x coordinate	F/R	N	M
YMIN	Minimum y coordinate	F/R	N	M
XMAX	Maximum x coordinate	F/R	N	M
YMAX	Maximum y coordinate	F/R	N	M

TABLE 29. Face bounding rectangle record example.

Column Name	Contents
ID	1
XMIN	Null
YMIN	Null
XMAX	Null
YMAX	Null
ID	2
XMIN	12.31
YMIN	37.97
XMAX	13.31
YMAX	37.99
ID	3
XMIN	14.54
YMIN	37.83
XMAX	14.58
YMAX	37.85

5.3.3 Feature class. A feature class is composed of a variety of tables containing geometric, topologic, and attribute data. Complex feature relationships can be defined. Some features may require a single feature and an associated primitive table, while others may need multiple tables linking features into a complex hierarchy. Attribute data may be extended by the use of related attribute tables (rat). A feature minimum bounding rectangle table may be included for each feature table as defined in TABLE 28. The feature class definition is provided by a product specification.

Constructing feature classes requires the use of feature tables. If necessary, the feature class definition may require the use of a feature join table to accurately construct the feature in relational form. Appendix C describes feature class relationships in greater detail.

5.3.3.1 Feature tables. A feature table consists of a feature identifier and one or more attribute columns. TABLE 30 defines feature table contents for non-compound features. If the coverage is tiled, the feature table maintains a triplet id column that identifies the tile id and primitive id that are related to the feature. This column is named after the primitive table it relates to, followed by "_ID." The tile id and the primitive id can be stored in separate columns. The tile id column is named TILE_ID. FROM_TO is an optional column used to provide directionality of a line feature with respect to its edge primitive(s). A FROM_TO value of 1 means the feature has the same orientation as its related primitive(s); a -1 means opposite orientation.

TABLE 30. Feature table definition.

Column Name	Description	Column Type	Key	Op/Man
ID	Row id	I	P	M
TILE_ID	Tile id	S	N	MT[2]
*_ID[1]	Primitive id	S/I/K	N	M
<Attribute n>	(attribute description)	Any	Any	M[4]
FROM_TO	Line feature orientation	S	N	O[3]

Notes: 1. The asterisk (*) indicates a placeholder for the primitive table name: EDG, FAC, END, CND, or TXT. If a join table exists, the TILE_ID, FROM_TO and *_ID columns will be found there.
2. If primitive id column is of type K, then no tile reference id column is required.
3. Optional for line feature tables only.
4. A minimum of one attribute is required for any table. If the table does not contain tile_id and/or *_id columns, an attribute will be mandatory for this column. Any additional attributes are optional.

All feature class definitions fall into one of five categories: area, line, point, text, and complex features.

a. Area feature. An area feature is composed of one or more face primitives. The primitive id column name will be FAC_ID. For instance, a vegetation feature will be defined by face primitives and will have various descriptive attributes (such as canopy closure, stem diameter size, and vegetation type category).

b. Line feature. Line features are composed of one or more edge primitives. The primitive id column name will be EDG_ID. A river feature could contain the following attributes: hydrographic category, depth, and width.

c. Point feature. A point feature class contains node primitives. The primitive id column name will be END_ID or CND_ID. For instance, a dam feature class may contain many attributes, such as height, length, and width.

d. Text feature. A text feature class is composed of a text primitive. The primitive id column name will be TXT_ID. The text feature usually contains additional attributes, such as text color, font, and font size.

e. Complex feature. Complex features can be constructed from any features. Complex features, however, cannot be recursively defined.

5.3.3.2 Feature join tables. Join tables are used to implement one-to-many relationships and/or many-to-many relationships between tables. They allow a variety of constructs to be made: between a feature table and a primitive table, between feature tables (for complex features), and between a feature table and an attribute table. A join table is used to relate one column in a table with a column in a second table. This is common when a 1:N relationship exists between two tables. Thus, for example, if a complex feature has four 1:N relations with four simple feature classes, four join tables will be used. The content and number of join tables depend on the coverage design and is defined in the Product Specification. The feature class schema table (fcs) documents all of the relationships for each feature class.

Unlike the feature table format that is dependent on the category (area, line, point, text, and complex), the same join table format is applicable to all feature classes. In the description below, a feature to primitive join relationship is modeled using the structure presented in TABLE 31. Together with the Feature Class Schema (FCS) table, this table structure can be utilized to represent other relationships; such as, relationships defined among features, among primitives, and so forth. Since joins are fully described in the FCS, there is usually no predetermined requirement on the join table columns. Furthermore, unless one of the tables in the join relation is a primitive table in a tiled coverage, the Tile_ID column is generally not necessary. TABLE 31 defines join table contents. The second column contains the key of the first table in the join. The name of this column is the table's name followed by "_ID." The third column contains the key of the second table in the join; it is similarly labeled using the table name and the "_ID" suffix. For instance, in a feature table (SDRPOINT.PFT) relating to a primitive (END), the key for the first table is SDRPOINT.PFT_ID and the key for the second table is END_ID. The feature class schema table is used to express the column names for the join table. The TILE_ID column is mandatory when tiled directories exist in the coverage and the join table relates a feature table and a primitive table. Refer to section 5.3.3.3 concerning these columns when tile directories exist in the coverage.

TABLE 31. Join table definition.

Column Name	Description	Column Type	Key	Op/Man
ID	Row id	I	P	M
*_ID[1]	Table 1 id	I	N	M
TILE_ID	Tile id	S	N	MT/O[3]
**_ID[2]	Table 2 id	I/S/K	N	M
FROM_TO	Line feature orientation	S	N	O[4]

Notes: 1. The asterisk (*) indicates a placeholder for the name of the first table in the join, usually a feature table name.

2. The (**) indicates a placeholder for the table name of the second table in the join, usually one of EDG, FAC, END, CND, or TXT.

3. If primitive id column is of type K, then no tile reference id column is required. Tile_id is also not required if this is a join between two feature tables, as in complex features, or between a feature table and an attribute table.

4. Optional for line features only.

5.3.3.3 Feature-to-primitive relations on tiled coverages. If the coverage is tiled, the feature or join table maintains a triplet id column that identifies the tile id and primitive id that are related to the feature. Alternatively, the tile id and the primitive id can be stored in separate columns. The triplet id column is named after the primitive table it relates to, followed by "_ID." The first field in the triplet id is null. The second field is the tile id (found in the tile reference coverage); the third field is the primitive row id in that tile. If the relation between the primitive and the feature is made using a join table, then this information will be stored in the join table unless it is stored in the feature table.

5.3.4 Coverage. Each coverage has one set of topological primitives and a collection of feature tables based upon these topological objects. All these tables are stored in one directory and are associated by file naming conventions (see TABLE 15). The coverage may contain a data dictionary for all feature tables in the coverage. An optional data quality table is allowed at the coverage level.

5.3.4.1 Coverage relationships. The general description of a coverage is stored in the coverage attribute table of its library. The relationships between the tables in a coverage are described by the mandatory feature class schema table.

5.3.4.2 Feature class schema table. A feature class schema table defines the feature classes that are contained within the coverage. Each record in the table specifies a feature class name, the name of the two tables involved in the join, and the names of the columns used in the join. The feature class name must be repeated to specify all the relationships in the feature class schema table. In the case of complex features, all relationships defining the component features must be specified with the feature class column referencing the complex feature class. This includes relationships between simple features and their primitives, even though these relationships are defined in other parts of the feature class schema table. Paragraph C.4.7.3 of Appendix C contains an example of relationships in a coverage with several simple features and a complex feature that consists of one point feature class and two line feature classes. The topological relationships need not be specified, since they are implied by table types. The feature class schema table must be used in conjunction with the TILEREF area feature table to fully define feature classes in tiled coverages.

If a key in the join is a compound key, the column names will be listed, separated by a backslash character (\). For example, a primary key composed of two columns would be specified as "NAME\TYPE." TABLE 32 defines feature class schema contents, and TABLE 33 is an example of a feature class schema table.

TABLE 32. Feature class schema definition.

Column Name	Description	Column Type	Key	Op/Man
ID	Row id	I	P	M
FEATURE_CLASS	Feature class name	T/L/M/N,8	N	M
TABLE1	The first table in a relationship	T/L/M/N,12	N	M
TABLE1_KEY	Join column in the first table	T/L/M/N,*	N	M
TABLE2	The second table in a relationship	T/L/M/N,12	N	M
TABLE2_KEY	Join column in the second table	T/L/M/N,*	N	M

TABLE 33. Feature class schema example.

Column Name	Contents
ID	1
FEATURE_CLASS	SDRPOINT
TABLE1	SDRPOINT.PFT
TABLE1_KEY	ID
TABLE2	END
TABLE2_KEY	ID
ID	2
FEATURE_CLASS	SDRPOINT
TABLE1	END
TABLE1_KEY	ID
TABLE2	SDRPOINT.PFT
TABLE2_KEY	ID
ID	3
FEATURE_CLASS	SDRAREA
TABLE1	SDRAREA.AFT
TABLE1_KEY	ID
TABLE2	FAC
TABLE2_KEY	ID
ID	4
FEATURE_CLASS	SDRAREA
TABLE1	FAC
TABLE1_KEY	ID
TABLE2	SDRAREA.AFT
TABLE2_KEY	ID

5.3.4.3 Value description table. A value description table relates to associated feature class tables within a coverage. VDTs are required when coded attribute values are implemented within a coverage. The two types of VDTs are integer VDT and character VDT. There will be no more than one of each per coverage. If a column in a feature table requires a VDT, then every unique coded value will have an entry in the VDT. Certain values (e.g., null or unknown) that are globally defined for all columns may not appear in a VDT, however, they will be documented in the product specifications. TABLE 34 defines the format of a VDT, and TABLE 35 provides an example.

TABLE 34. Value description table definition.

Column Name	Description	Column Type	Key	Op/Man
ID	Row id	I	P	M
TABLE	Feature table name	T/L/M/N,12	N	M
ATTRIBUTE	Column name	T/L/M/N,n	N	M
VALUE	Unique attribute value	I/S/T/L/M/N,n	N	M
DESCRIPTION	Attribute description	T/L/M/N,*	N	M

TABLE 35. Integer value description record example.

Column Name	Contents
ID	1
TABLE	SDRPOINT.PFT
ATTRIBUTE	MCP
VALUE	0
DESCRIPTION	UNKNOWN
ID	2
TABLE	SDRPOINT.PFT
ATTRIBUTE	MCP
VALUE	18
DESCRIPTION	CONCRETE
ID	3
TABLE	SDRPOINT.PFT
ATTRIBUTE	MCP
VALUE	23
DESCRIPTION	EARTHEN
ID	4
TABLE	SDRPOINT.PFT
ATTRIBUTE	HYC
VALUE	0
DESCRIPTION	UNKNOWN
ID	5
TABLE	SDRPOINT.PFT
ATTRIBUTE	HYC
VALUE	6
DESCRIPTION	NON-PERENNIAL
ID	6
TABLE	SDRPOINT.PFT
ATTRIBUTE	HYC
VALUE	10
DESCRIPTION	TIDAL

5.3.5 Library. The function of a VPF library is to organize collections of coverages that pertain to one geographic region. Each library must manage its own spatial extent. VPF uses a minimum bounding rectangle to define this geographic extent. VPF also supports a coverage, the library reference coverage, which describes the library region in a graphic manner. All reference coverages must be spatially registered and be in the same coordinate system.

Within each library, there are three mandatory tables and seven optional tables. The library header table defines the contents of the library. The geographic reference table contains information pertaining to the geographic location of the library. A coverage attribute table provides a list of and descriptions for the coverages contained in the library.

If the library is tiled, there will be an untiled tile reference coverage. There will also be an untiled library reference coverage that gives a general overview of the information contained in the library. In addition, it is possible to include a data quality coverage. The format for the data quality and library reference coverage follow the same rules as any normal VPF coverage. See appendix E for information concerning recommended data quality coverage contents. Multiple records in these tables are used to describe multiple sources, updates and maintenance issues.

5.3.5.1 Library header table. The library header table contains information identifying the library, general information about the contents, security, and source. TABLE 36 describes the group of entities that compose the library header table.

TABLE 36. Library header table.

Column Name	Description of Content	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
PRODUCT_TYPE	Series designator of product type	T,12	N	M
LIBRARY_NAME	Name of library	T,12	N	M
DESCRIPTION	Text description of library	T,100	N	M
DATA_STRUCT_CODE	Highest level code for the library 5 = Level 0 topology 6 = Level 1 topology 7 = Level 2 topology 8 = Level 3 topology	T	N	M
SCALE	Source scale of the library (i.e. 200) using the representative fraction denominator	I	N	M
SOURCE_SERIES	Series designator (e.g. 1501)	T,15	N	M
SOURCE_ID	Source id - number or name that when used in conjunction with the series and edition will identify a unique source	T,30	N	M
SOURCE_EDITION	Source edition number	T,20	N	M
SOURCE_NAME	Full name of source document	T,100	N	M
SOURCE_DATE	Significant date. A designed date that most accurately describes the basic date of the product for computation of the probable obsolescence date. It can be the compilation date or the revision date or other depending on the product and circumstances	D	N	M

SECURITY_CLASS	Security classification of the source
	T=TOP SECRET
	S=SECRET
	C=CONFIDENTIAL
	R=RESTRICTED (or alternatively "FOR OFFICIAL USE ONLY" administrative classification only)
	U=UNCLASSIFIED
		T	N	M

DOWNGRADING	Originator's permission for downgrading required (yes or no)	T,3	N	M
DOWNGRADING_DATE	Date of downgrading (null if answer to previous entity is yes)	D	N	M
RELEASABILITY	Releasability restrictions	T,20	N	M

5.3.5.2 Geographic reference table. This table (TABLE 37) contains four groups of fields that define the geographic parameters of the library. These field groups are the geographic parameters, projections, registration points table name, and diagnostic points table name. The geographic parameters in this table serve two purposes. Firstly, they are for descriptive documentation of the coordinate reference system used in the database. Secondly, they allow multiple separately published libraries to be inversely projected into a common coordinate system for display and query. If the database is maintained in unprojected geographic coordinates, the projection code and its corresponding projection parameters need not be included in the table. (Refer to APPENDIX G for valid values in datum codes, projection codes, and unit of measure codes.)

TABLE 37. Geographic reference table.

Column Name	Description of Contents	Field Type	Key Type	Op/Man
ID	Row id	I	P	M
DATA_TYPE	Type of data in the library	T,3	N	M
UNITS	Units of measure code for coordinates in library	T,3	N	M
ELLIPSOID_NAME	Name of ellipsoid of the library	T,15	N	M
ELLIPSOID_DETAIL	Details about library ellipsoid including ellipsoid code	T,50	N	M
VERT_DATUM_NAME	Name of vertical reference	T,15	N	M
VERT_DATUM_CODE	Code of vertical datum reference	T,4	N	M
SOUND_DATUM_NAME	Name of sounding datum	T,15	N	M
SOUND_DATUM_CODE	Code of sounding datum reference	T,4	N	M
GEO_DATUM_NAME	Name of geodetic datum	T,15	N	M
GEO_DATUM_CODE	Code of geodetic datum	T,4	N	M
PROJECTION_NAME	Name of the projection	T,20	N	M
PROJECTION_CODE	Code of the projection if any (null if geographic coordinates)	T,2	N	O
PARAMETER1	Projection parameter 1	F	N	O
PARAMETER2	Projection parameter 2	F	N	O
PARAMETER3	Projection parameter 3	F	N	O
PARAMETER4	Projection parameter 4	F	N	O
FALSE_ORIGIN_X	False Easting origin of projection	F	N	O
FALSE_ORIGIN_Y	False Northing origin of projection	F	N	O
FALSE_ORIGIN_Z	False origin for Z values	F	N	O
REG_PT_TABLE	Registration point table	T,12	N	O
DIAG_PT_TABLE	Diagnostic point table	T,12	N	O

5.3.5.3 Coverage attribute table. This table contains the coverage name and topology level for each coverage in the library. Only those coverages which actually exist within a specific library will be listed in that library's coverage attribute table. The topological level associated with each coverage determines the nature of geometric and topological information available on that coverage. TABLE 38 defines coverage attribute table entities. TABLE 39 is an example of a coverage attribute table.

TABLE 38. Coverage attribute table definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	U	M
COVERAGE_NAME	The coverage name	T,8	P	M
DESCRIPTION	Description string	T,*	N	M
LEVEL	The topologic level	I	N	M

TABLE 39. Coverage attribute table example.

Column Name	Contents
ID	1
COVERAGE_NAME	SLP
DESCRIPTION	Surface configuration
LEVEL	3
ID	2
COVERAGE_NAME	VEG
DESCRIPTION	Vegetation
LEVEL	3
ID	3
COVERAGE_NAME	SMC
DESCRIPTION	Surface materials
LEVEL	3
ID	4
COVERAGE_NAME	SDR
DESCRIPTION	Surface drainage
LEVEL	3
ID	5
COVERAGE_NAME	TRN
DESCRIPTION	Transportation
LEVEL	3
ID	6
COVERAGE_NAME	OBS
DESCRIPTION	Obstacles
LEVEL	3

5.3.5.4 Tile reference coverage. A library may contain tiled partitions and will require a tile reference coverage, TILEREF, with associated area feature and attribute tables. Thus, if the tile reference coverage does not exist in a library, then no tiling scheme exists. The tile reference coverage consists of a set of faces identifying the tiles that the library uses to subdivide the region of interest. Tile attributes are used to optimize retrievals. At the library level, the tile geometry is simple, describing the location of the tile boundaries.

5.3.5.4.1 Tile attributes. The tile attributes are located in the area feature table of the tile reference coverage. This table maintains a unique id and its associated tile directory name. This directory name may contain a nested directory list or path name, relative to the coverage level. If a path name is required, the separator character will be a backslash ('\'). Additional columns may exist for every feature class or coverage located in the library, but this is optional. The id of the area feature table will be used in the tile and primitive id for each coverage feature table (see section 5.3.3.1).

TABLE 40 defines the tile reference coverage attribute columns. TABLE 41 is an example of a tile reference coverage area feature table.

TABLE 40. Tile reference area feature table definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
TILE_NAME	Tile name	T,n (n<=64)	U	M
FAC_ID	Face id	S/I/K	N	M

TABLE 41. Tile area feature record example.

Column Name	Contents
ID	1
TILE_NAME	M\J\12
FAC_ID	5
ID	2
TILE_NAME	M\J\22
FAC_ID	7
ID	3
TILE_NAME	M\J\23
FAC_ID	2
ID	4
TILE_NAME	M\J\24
FAC_ID	3

5.3.5.5 Registration point table. The registration point table includes columns for the latitude, longitude and elevation of ground points selected as registration points. This table also includes columns for the corresponding x, y and z table coordinates. Each registration point in this table is identified by a row id and a registration point id. The geographic reference table contains an optional column containing the registration point table name, which carries the file extension .rpt. (See TABLE 42.)

TABLE 42. Registration point table.

Column Name	Description of Contents	Field Type	Key Type	Op/Man
ID	Row id	I	P	M
REG_PT_ID	Registration point id	I	N	M
REG_LONG	Longitude of registration point	F	N	M
REG_LAT	Latitude of registration point	F	N	M
REG_Z	Elevation of registration point	F	N	M
REG_table_X	X table coordinate of control pts	F	N	M
REG_table_Y	Y table coordinate of control pts	F	N	M
REG_table_Z	Z table coordinate of control pts	F	N	M

5.3.5.6 Diagnostic point table. The diagnostic point table includes columns for the latitude, longitude and elevation of ground points selected as diagnostic points. This table also includes columns for the corresponding x, y and z table coordinates. Each diagnostic point in this table is identified by a row id and a diagnostic point id. The geographic reference table contains an optional column containing the diagnostic point table name, which carries the file extension .dpt. (See TABLE 43.)

TABLE 43. Diagnostic point table.

Column Name	Description of Contents	Field Type	Key Type	Op/Man
ID	Row id	I	P	M
DIAG_PT_ID	Diagnostic point id	I	N	M
DIAG_LONG	Longitude of diagnostic point	F	N	M
DIAG_LAT	Latitude of diagnostic point	F	N	M
DIAG_Z	Elevation of diagnostic point	F	N	M
DIAG_table_X	X table coordinate of diagnostic pts	F	N	M
DIAG_table_Y	Y table coordinate of diagnostic pts	F	N	M
DIAG_table_Z	Z table coordinate of diagnostic pts	F	N	M

5.3.6 Database. Information that applies to the whole collection of data belongs at the database level. Structurally, a database consists of a set of libraries. It is possible to include a data quality table.

There are two mandatory tables: the library attribute table and the database header table. These two tables are described below. Multiple records in each table are used to describe multiple sources, updates, and maintenance issues.

5.3.6.1 Library attribute table. The library attribute table contains the name and extent for each library in the database. The library minimum bounding rectangle shall be in latitude and longitude (decimal degrees). TABLE 44 defines library attribute entities. TABLE 45 is an example of a library attribute table.

TABLE 44. Library attribute table entity definitions.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	U	M
LIBRARY_NAME	Library name	T/L/M/N,8	P	M
XMIN	Westernmost Longitude	F/R	N	M
YMIN	Southernmost Latitude	F/R	N	M
XMAX	Easternmost Longitude	F/R	N	M
YMAX	Northernmost Latitude	F/R	N	M

TABLE 45. Library attribute table example.

Column Name	Contents
ID	1
LIBRARY_NAME	NOAMER
XMIN	-97.750000
YMIN	31.167000
XMAX	-97.667000
YMAX	31.250000

5.3.6.2 Database header table. The database header table (TABLE 46) contains information that defines database content and security information.

TABLE 46. Database header table definition.

Column Name	Description of Contents	Column Type	Key Type	Op/Man
D	Row id	I	P	M
VPF_VERSION	VPF version number	T,10	N	M
DATABASE_NAME	Directory name of the database	T,8	N	M
DATABASE_DESC	Text description of the database	T,100	N	M
MEDIA_STANDARD	Media standard used for the database	T,20	N	M
ORIGINATOR	Text for title and address of originator (a backslash "\" is used as a line separator)	T,50	N	M
ADDRESSEE	Text for title and address of addressee (a backslash "\" is used as a line separator)	T,100	N	M
MEDIA_VOLUMES	Number of media volumes comprising the database	T,*	N	M
SEQ_NUMBERS	Sequential number(s) for each media volume in this database	T,*	N	M
NUM_DATA_SETS	Number of libraries within database	T,*	N	M
SECURITY_CLASS	Security classification of database (the highest security classification of the transmittal including all datasets within the database) T=TOP SECRET S=SECRET C=CONFIDENTIAL R=RESTRICTED (or alternatively "FOR OFFICIAL USE ONLY") U=UNCLASSIFIED	T	N	M
DOWNGRADING	Originator's permission for downgrading required (yes or no)	T,3	N	M
DOWNGRADE_DATE	Date of downgrading	D	N	M
RELEASABILITY	Releasability restrictions	T,20	N	M
OTHER_STD_NAME	Free text, note of other standards compatible with this database	T,50	N	O
OTHER_STD_DATE	Publication date of other standard	D	N	O
OTHER_STD_VER	Other standard amendment number	T,10	N	O
TRANSMITTAL_ID	Unique id for this database	T,*	N	M
EDITION_NUMBER	Edition number for this database	T,10	N	M
EDITION_DATE	Creation date of this database	D	N	M

5.3.7 Data quality. The data quality table may be stored at the database, library, or coverage level. It contains information on the completeness, consistency, date status, attribute accuracy, positional accuracy, and other miscellaneous quality information. It also contains information about the source from which the geographic data was derived. Lineage information should be included in the associated narrative table, named "LINEAGE.DOC." While the contents and location of a data quality table within a VPF database are product specific, it is highly recommended that at least one table exist at the library level. TABLE 47 defines the contents of the data quality table. APPENDIX E contains more information about storing data quality information in VPF.

TABLE 47. Data quality table definition.

Column	Name Description of Contents	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
VPF_LEVEL	Either DATABASE or LIBRARY or COVERAGE	T,8	N	M
VPF_LEVEL_NAME	Directory name of database or library or coverage	T,8	N	M
FEATURE_COMPLETE	Feature completeness percent	T,*	N	M
ATTRIB_COMPLETE	Attribute completeness percent	T,*	N	M
LOGICAL_CONSIST	Logical consistency	T,*	N	M
EDITION_NUM	Edition number	T,8	N	M
CREATION_DATE	Date of creation	D	N	M
REVISION_DATE	Date of revision	D	N	M
SPEC_NAME	Name of product specification (e.g. DCW)	T,*	N	M
SPEC_DATE	Date of product specification	D	N	M
EARLIEST_SOURCE	Date of earliest source	D	N	M
LATEST_SOURCE	Date of latest source	D	N	M
QUANT_ATT_ACC	Standard deviation of quantitative attributes	T,*	N	O
QUAL_ATT_ACC	Percent reliability of qualitative attributes	T,*	N	O
COLLECTION_SPEC	Name of collection specification	T,*	N	M
SOURCE_FILE_NAME	Name of included source file	T,12	N	O
ABS_HORIZ_ACC	Absolute horizontal accuracy of database or library or coverage	T,*	N	M
ABS_HORIZ_UNITS	Unit of measure for absolute horizontal accuracy	T,20	N	M
ABS_VERT_ACC	Absolute vertical accuracy of database or library or coverage	T,*	N	M
ABS_VERT_UNITS	Unit of measure for absolute vertical accuracy	T,20	N	M
REL_HORIZ_ACC	Point to point horizontal accuracy of database or library or coverage	T,*	N	M
REL_HORIZ_UNITS	Unit of measure for point to point horizontal accuracy	T,20	N	M
REL_VERT_ACC	Point to point vertical accuracy of database or library or coverage	T,*	N	M
REL_VERT_UNITS	Unit of measure for point to point vertical accuracy	T,20	N	M
COMMENTS	Miscellaneous comments - free text	T,*	N	M

Multiple records in each table are used to describe multiple sources, updates, and maintenance issues.

5.3.8 Narrative table. The narrative table (TABLE 48) will contain any descriptive information concerning its associated table. The contents of the TEXT column will be product specific.

TABLE 48. Narrative table definition.

Column Name	Description	Column Type	Key Type	Op/Man
ID	Row id	I	P	M
TEXT	Text information	T/L/M/N,*	N	M

5.3.9 Names reference coverage. The names reference coverage is optional at the library level. The names reference coverage must maintain the same bounding rectangle as the other library coverages (tile and library reference coverages). Other feature classes may exist, but are not required.

The names reference coverage will contain at least the following: a point feature table; an entity node primitive table; and a thematic index created on a fixed-length text column in the point feature table. The column's name will be "PLACE_NAME."

5.4 VPF encapsulation. Encapsulation defines the structure of data fields and the grouping of these fields. A simple table-oriented data encapsulation system is defined for VPF which allows the use of binary coded numeric data as well as text data and which uses references to identify the location of data elements within numerous related tables stored in files.

5.4.1 Table definition. A VPF table consists of a header and at least one record. Records can be of variable length; indexes can be used to directly access the files as needed. A table will begin with a header defining the table contents, followed by a series of records (rows). Where table records are not of fixed length, an external index containing record offsets and length (in bytes) will be used to provide direct access (see FIGURE 19).

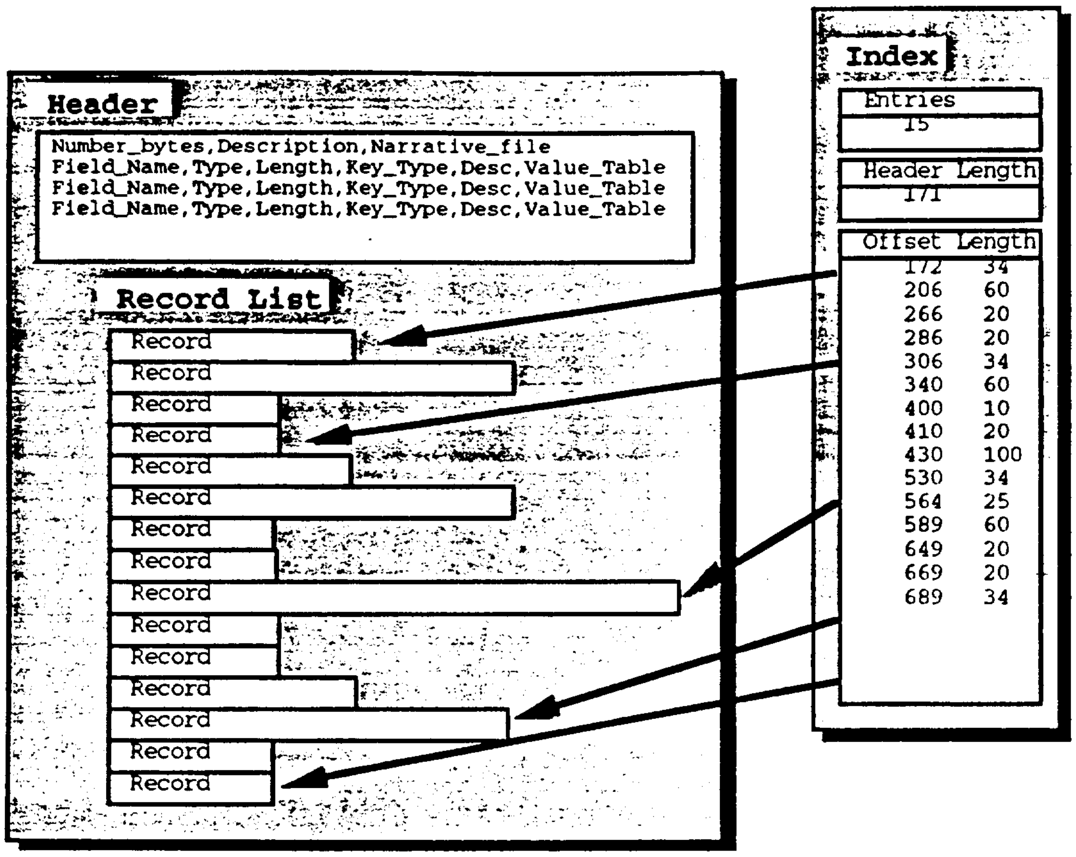

Header
Number_bytes,Description,Narrative_file
Field_Name,Type,Length,Key_Type,Desc,Value_Table
Field_Name,Type,Length,Key_Type,Desc,Value_Table
Field_Name,Type,Length,Key_Type,Desc,Value_Table
Record List
Record
Index
Entries
15
Header Length
171
Offset Length
172 34
206 60
266 20
286 20
306 34
340 60
400 10
410 20
430 100
530 34
564 25
589 60
649 20
669 20
689 34

FIGURE 19. Table structure.

5.4.1.1 Header. A table's header (described in TABLE 49) is composed of two parts. The first part is a 4-byte integer that indicates the length of the following text string, which defines the table. To accommodate different hardware architecture, after the length field a byte order field may be inserted. A value of "L" in the byte order field indicates a least-significant-byte-first encoding; an "M" indicates a most-significant-byte-first encoding. Least-significant-byte-first encoding is assumed if the byte order field is not present.

The second part, the header definition text string, contains three components, each of which is separated by a semicolon. The semicolon (;) indicates the end of the component.

TABLE 49. Header field definitions.

Field	Description	Column Type
Header length	Length of ASCII header string (i.e., the remaining information after this field)	I
Byte order	Byte order in which table is written: L - least-significant-first M - most-significant-first	T,1
	(semicolon separator)	T,1
Table description	Text description of the table's contents	T/L/M/N,n (n≤80)
	(semicolon separator)	T,1
Narrative table	An optional narrative file which contains miscellaneous information about the table	T/L/M/N,n (n≤12)
	(semicolon separator)	T,1
Column definitions	The following fields repeat for each column contained in the table:	
Name	Name of the column	T/L/M/N,n (n≤16)
	(equal sign separator)	T,1
Data type	One of the field types found in table 56	T,1
	(comma separator)	T,1
Number[1]	Number of elements	T,n (n≤3)
	(comma separator)	T,1
Key type	Key type	T,1
	(comma separator)	T,1
Column description	Text description of the column's meaning	T/L/M/N,*
	(comma separator)	T,1
Value description table name	Name of an associated value description table	T/L/M/N,n (n≤12)
	(comma separator)	T,1
Thematic index	Name of thematic indexes.	T/L/M/N,n (n≤12)
	(comma separator)	T,1
Narrative table name	Name of an associated narrative table	T/L/M/N,n (n≤12)
	(comma separator)	T,1
End of column	(colon separator)	T,1
	... (repeat for each column)	
End of header	(semicolon separator)	T,1

Note 1. This field contains the number of occurrences of the data type specified, not the number of bytes. For example, if there is only one integer value in the field, the header will

contain the number "1" in that field. For text fields only, the value indicates the maximum of bytes allowed for that column. For example, if a maximum of 12 characters are allowed in the field, then the number of elements is specified as "12". The number of bytes specified for a particular text field are shown in subsequent tables in this specification.

The first component is the table description. The second component is a file reference to a narrative text, if available. If no narrative file exists, the dash symbol (-) is used for the file reference. The final component is the column definition substring. The column definitions are separated by colons (:), which indicate the end of the subcolumn definition. If an entry is not applicable to the field (i.e., a thematic index does not exist), the dash symbol (-) is used to indicate a null value. Trailing null field entries need not be included. For clarity in documentation, these trailing null fields should be listed, however. For each column in the table, there will be:

a. Column name, followed by an equal sign (=)
b. Data type indicator, followed by a comma (,)
c. Number of data type elements, followed by a comma
d. Key type indicator, followed by a comma
e. Column description, followed by a comma
f. Value description table name, if any, followed by a comma
g. Thematic index name, if any, followed by a comma
h. Column narrative file name, if any, followed by a comma

The character used as a separator for a particular field will not appear in that field except as a separator. For example, the Table Description header field will never include an imbedded semi-colon because a semi-colon is the specified separator. However, this field may legally contain imbedded colons or commas because neither of these characters are the separator for Table Description.

TABLE 50 displays an example of a text header for an area feature table. For presentation purposes, each component in the table definition string is listed on a separate line in TABLE 51. No new line, space, or tab character should be inserted after the field and component separators in the actual table definition string. Furthermore, the example does not show the 4-byte definition string length and the byte order character.

TABLE 50 shows a header definition string for a Surface Drainage Area area feature table. A narrative file, SDR.DOC, is attached to this table in the VPF database. The table has 13 columns, with the column ID being the primary key column.

TABLE 50. Text header example.

```
Surface Drainage Area;
SDR.DOC;
ID=I,P,Row ID,-,-,-,:
F_CODE=T,5,N,FACS Code,CHAR.VDT,F_CODE.ATI,F_CODE.DOC,:
RGC=I,1,N,Railroad Gauge Category,INT.VDT,-,-,:
HYC=I,1,N,Hydrographic Category,INT.VDT,-,-,:
HFC=I,1,N,Hydrographic Form Category,INT.VDT,-,-,:
EXS=I,1,N,Existence Category,INT.VDT,-,-,:
WID=F,1,N,Width (meters),-,-,-,:
WV1=I,1,N,Water Velocity Average (m/sec),INT.VDT,-,-,:
WDA=I,1,N,Water Depth Average (meters),INT.VDT,-,-,:
MCP=I,1,N,Material Composition Primary,INT.VDT,-,-,:
DVR=I,1,N,Dense Bank Vegetation Right,INT.VDT,-,-,:
DVL=I,1,N,Dense Bank Vegetation Left,INT.VDT,-,-,:
BGR=I,1,N,Bank Gradient (Slope) Category Right Bank (%),-,-,-,:;
```

5.4.1.2 Record list. The body of data contained within the table is the record list; the header and the table index serve only to define the contents and provide effective access to this list. These records can be of fixed or variable length, as needed.

5.4.1.3 Variable-length index file. The variable-length index is a separate file that is mandatory when a VPF table contains variable-length records. As shown in TABLE 51, the file has two parts: a header and a data array. Each entry in the data array relates to a record in the VPF table.

TABLE 51. Components of variable-length index file.

File Component	Content of Component	Data Type	Field Length
Header	Number of entries (N) in index (which also matches the number of records in the associated table)	Integer	4 bytes
	Number of bytes in VPF table header	Integer	4 bytes
Data array	A two-dimensional array of N records	Integer	8N bytes

The data array identifies the location of every record in the variable-length file by containing the following entries for each record:

[n][0]	=	Byte offset from beginning of file
[n][1]	=	Number of bytes in table record

where n is an integer from 1 to N. The term byte offset refers to a location with respect to the beginning of a file. The first byte of a file has an offset of zero.

Thus, if the software requires the location of record 45 in a VPF table, the index file can be used to locate the exact position of the record without sequentially searching for the match. The entry for record 45 in the variable-length index would indicate the byte offset in the VPF table to the position of record 45 and the number of bytes in record 45.

5.4.2 Spatial index files. For each primitive (face, edge, entity node, connected node, and text), there can exist a spatial index file that will accelerate queries by software. Although these files are optional, they are recommended, especially for large libraries. These indexes are indirectly created on the coordinate column or the minimum bounding rectangle of each primitive; appendix F contains more information.

The format of the spatial index is as follows:

a. Header record. The header will contain one integer defining the number of primitives (NUMPRIM) and another integer defining the number of nodes (NNODE) in the index. Between the two integer fields are four (xmin, ymin, xmax, ymax) short floating point coordinates defining the minimum bounding rectangle. TABLE 52 shows the layout for the spatial index file header record.

TABLE 52. Spatial index file header record layout.

Byte Offset	Width	Type	Description
0	4	Integer	Number of primitives
4	4	Floating point	MBR x1
8	4	Floating point	MBR y1
12	4	Floating point	MBR x2
16	4	Floating point	MBR y2
20	4	Integer	Number of nodes in tree

b. Bin array record. This record is a two-dimensional array the length of which is NNODE, described in the header record. The structure of this record is shown in TABLE 53. This array maintains a long integer offset that points to the beginning of the bin data record and a long integer primitive count for each bin. The offset for the first bin always has a value of zero.

For bins that contain no primitives, the value assigned to the count variable is zero and the offset value is zero.

TABLE 53. Structure of the bin array record.

Byte Offset	Width	Type	Description
HDR + n * 8	4	Integer	Offset of primitive list for node n
HDR + n * 8 + 4	4	Integer	Count in integer units

Note: n is {0 ... number of nodes-1}; the n value for the first node is 0. HDR is the length of the index file header record.

c. Bin data record. There are NUMPRIM records where each record contains four 1-byte integers defining the MBR for the primitive and one long integer (4 bytes) for the primitive id. These primitive ids point into the associated primitive table. TABLE 54 shows the structure of the bin data record.

TABLE 54. Structure of the bin data record.

Byte Offset	Width	Type	Description
HDR+BIN+OS + c * 8 + 0	1	byte	MBR x1
HDR+BIN+OS + c * 8 + 1	1	byte	MBR y1
HDR+BIN+OS + c * 8 + 2	1	byte	MBR x2
HDR+BIN+OS + c * 8 + 3	1	byte	MBR y2
HDR+BIN+OS + c * 8 + 4	4	int	Primitive id

Note: c is {0 ... number of primitives for a node - 1}; the c value for the first primitive is 0. HDR is the length of the index file header record. BIN is the summed length of all the bin array records. OS is the value of the offset variable in the corresponding bin array record (as shown in the byte offset calculation in TABLE 53).

5.4.3 Thematic index files. A thematic index may be created for any column in a table. There are two types of indexes, depending on the data content in a column: an inverted list thematic index or a bit array thematic index.

For categorical data or data with few distinct values, such as soil polygons where numerous polygons are assigned soil class designations from a relatively small number of classes, an inverted list is used. One entry in the index file is created for each distinct value in the column; correspondingly, a list of table record ids is stored with the value.

If the data in a column is all unique, especially in the case of an index for character strings, a bit array can be stored for each unique byte/character in the column. Each bit in the bit array represents a row in the indexed table. An 'ON' bit at a particular position means that the corresponding row in the table contains a specific byte/character pattern.

The character string form of the thematic index is used for names placement index implementations.

The thematic index file may be partitioned into three data groups: a fixed-length header, a variable number of index (or directory) entries (another index within an index), and a set of rows. Each row contains VPF record ids stored either as a list or as a bit array. Each directory entry describes the element being indexed and the location of the row containing the list (or set) of record ids related to the element.

a. Header. The thematic index header contains 60 bytes of information that pertain to the type of index it is, the table it is associated with, and the column in that table. The layout of the header record is shown in TABLE 55. An optional ordering of the entries in the index directory can be specified using the ordering flag at offset 56 in the index header. An "S" in the ordering flag indicates an ascending sort order in the index directory. Entries in the directory are assumed not to have any specific ordering otherwise.

b. Index directory. The index directory contains repeating records for each distinct element being indexed. The structure of an index directory record is shown in TABLE 56. The number of entries is stored in the header record. Entries in the index directory give an offset at which the actual data are stored. There is also a count indicating the number of items maintained for a particular index value. If the count field in an index directory entry has a value of zero, the offset field contains the actual data; otherwise, the offset field contains an indirect address for the indexed data.

TABLE 55. Thematic index file header record layout.

Byte Offset	Width	Type	Description
0	4	Integer	Combined length of the index file header and the index directory.
4	4	Integer	Number of directory entries. This is the number of items being indexed by a particular index file.
8	4	Integer	Number of rows in the data table from which this index file was derived.
12	1	Character	Type of index file; either "I" for inverted list index, or "B" for bit array index.
13	1	Character	Type of the data element being indexed; one of: "I"—4-byte integer "T"—character string "S"—2-byte integer "F"—4-byte floating point "R"—8-byte floating point
14	4	Integer	Number of data elements comprising one directory entry. This field will usually have a value of 1; an exception is a thematic index built on a text field.
18	1	Character	Type specifier for the data portion of an index file. Record ids in inverted list index can be stored by using either a 2-byte integer (type "S") or a 4-byte one (type "I"). Bit array index files always use type "S."
19	12	Character	The name of the VPF table from which the index file has been derived; no path information is included.
31	25	Character	The name of the column in the VPF table from which index entries have been pulled.
56	1	Character	Ordering flag ("S" indicates an ascending order in the index directiory; no specific sort order otherwise).
57	3	Character	Unused and reserved.

TABLE 56. Structure of index directory record.

Byte Offset	Width	Type	Description
HDR +n*(d+8)	d	Character string 2-byte integer 4-byte integer 4-byte floating point 8-byte floating point	The element being indexed.
HDR +n*(d+8)+d	4	Integer	The offset from the beginning of the file to the location of the row associated with this index entry.
HDR +n*(d+8)+d+4	4	Integer	The number of indexed records associated with this entry. (For bit array index files, this is the number of bytes.)

Note: n is {0 ... number of index entries-1), and d = size of (indexed type). HDR is the length of the index file header record.

c. Index data. For each index entry there exists a data record consisting of either a list of row ids from the indexed file or a bit array.

The following example shows an inverted list thematic index created on a feature table column of data type 'S' (short integer). The name of the column is USE_CODE. The name of the table is CULAREA.AFT. The index table header shown in TABLE 57 is thus,

TABLE 57. Thematic index header example.

Byte Offset	Width	Type	Value
0	4	Integer	90 (header length + index directory length)
4	4	Integer	3 (number of directory entries)
8	4	Integer	293 (number of indexed rows)
12	1	Character	I (inverted list)
13	1	Character	S (short integer source data)
14	4	Integer	1 (data length is 1)
18	1	Character	S (short integer indexed id)
19	12	Character	CULAREA.AFT
31	25	Character	USE_CODE
56	1	Character	S (sorted index directory)
57	3	Character	• •

The value at byte offset 60 is the value of the element being indexed. The number of rows from the table CULAREA.AFT is contained at address HDR+n*(d+8)+d+4 (in this case at byte offset 66). The first entry in the directory has a USE_CODE value of 2, and there are 5 rows that contain the value. The CULAREA.AFT rows can be found at address 90. Another index directory entry starts at offset 70. This entry has a count of zero, indicating that the offset field contains the row number for the CULAREA.AFT table. The index directory shown in TABLE 58 is thus,

TABLE 58. Thematic index directory example.

Byte Offset	Width	Type	Value	
60	2	Short Integer	2	(index value)
62	4	Integer	90	(offset)
66	4	Integer	5	(count)
70	2	Short Integer	3	
72	4	Integer	20	(direct row id)
76	4	Integer	0	(count of "zero")
80	2	Short Integer	4	
82	4	Integer	100	
86	4	Integer	4	

The index data shown in TABLE 59 are thus,

TABLE 59. Thematic index data example.

Byte Offset	Count	Row numbers
90	5	8 9 10 11 12
100	4	22 23 24 25

5.4.3.1 Feature index. Feature indices may be created for any VPF coverage. These are join indices that have been developed to enhance processing of complex queries in relational databases. This is particularly significant in tiled VPF coverages with a number of feature classes. The VPF Standard specifies a set of join indices for feature/primitive joins. One feature join index can be defined for each of the five primitive types in a coverage. For example, an edge feature join index, edg.fit, can be defined for edge primitives and the corresponding line and complex features that reference those primitives.

Feature join indices are optional. Feature indices are composed of: (a) a feature class attribute table (FCA) and (b) a number of feature index tables (FIT). Feature index tables allow quick retrieval of feature information when given a selected primitive. They bypass the normal relational operations usually required and prestore the results of feature-to-primitive and primitive-to-feature relations. As indices, they may be deleted at any time and the relationships between tables will be maintained provided the associated join tables have been defined. Any updates to the data will cause the indices to be rebuilt. The feature index tables may not be referenced in the FCS. There could be one FIT for each primitive type in a coverage. For a given coverage, if a feature index is defined for a primitive type, all feature classes associated with that primitive type must be indexed. All FCA and FIT's reside at the coverage level.

A VPF coverage can optionally contain a Feature Class Attribute table (FCA). This table should minimally have the following columns: a feature class ID column (ID), a feature class name column (FCLASS), the feature type (TYPE) and a feature class description column (DESCR). A feature class attribute table definition example is shown in TABLE 60.

TABLE 60. Feature class attribute table definition.

Column Name	Description	Field Type	Key Type	Op/Man
ID	Row ID	I	P	M
FCLASS	Feature class name	T/L/M/N,8	U	M
TYPE	Feature type (P-point, L-line, A-area, T-text, C-complex)	T,1	N	M
DESCR	Description	T/L/M/N,*	N	M

Every primitive/feature reference, both directly and indirectly, as in the case of complex features, results in one entry in the appropriate FIT for that primitive and the corresponding feature. If a feature is composed of multiple primitives, each of those feature/primitive relationships is recorded. Conversely, if a primitive is applicable for more than one feature, multiple relationships are similarly maintained. When a primitive is referenced by a complex feature via an intermediate feature, the relationship between the primitive and the complex feature, as well as that between the primitive and the intermediate feature, are recorded in the FIT's.

Feature Index Tables (FIT) are made up of two compound keys, the feature class id (FC_ID) and the feature id (FEATURE_ID) to properly identify an individual geographic feature, and the tile id (TILE_ID) and primitive id (PRIM_ID) for a primitive. Available FIT names are: EDG.FIT, CND.FIT, END.FIT, FAC.FIT and TXT.FIT. An example of a feature index table definition is shown in TABLE 61.

TABLE 61. Feature index table definition.

Column Name	Description	Field Type	Key Type	Op/Man
ID	Row ID	I	P	M
PRIM_ID	Primitive id (foreign key to primitive table)	I	N	M
TILE_ID	Tile reference id	S	N	MT
FC_ID	Feature class id (foreign key to FCA)	I	N	M
FEATURE_ID	Feature id (foreign key to feature table)	I	N	M

5.4.4 Allowable field types. The field types depicted in TABLE 62 are allowed and provide the ability to encode any data set. All variable-length types include a count item "n" (as depicted in TABLE 56) preceding the data. The count is a 4-byte integer. This count item contains the number of bytes in text strings and the number of tuples in coordinate strings.

TABLE 62. Allowable field types.

Abbrv	Column Type	Null/No Value	Length (bytes)
T,n	Fixed-length text	"N/A"	n
T,*	Variable-length text	Zero length	n + 4
L,n	Level 1 (Latin 1 - ISO 8859) Fixed-length text	"N/A"	n
L*	Level 1 (Latin 1 - ISO 8859) Variable-length text	zero length	n+4
N,n	Level 2 (Full Latin - ISO 6937) Fixed-length text	"N/A"	n
N*	Level 2 (Full Latin - ISO 6937 Variable-length text	zero length	n+4
M,n	Level 3 (Multilingual - ISO 10646) Fixed-length text	"N/A"	n
M*	Level 3 (Multilingual - ISO 10646) Variable-length text	zero length	n+4
F	Short floating point	NaN (not a number)	4
R	Long floating point	NaN	8
S	Short integer	bit pattern 10000000 00000000	2
I	Long integer	bit pattern 10000000 00000000 00000000 00000000	4
C,n	2-coordinate array, short floating point	Both coordinates equal to null	8n
C,*	2-coordinate string	Length = 0	8n + 4
B,n	2-coordinate array, long floating point	Both coordinates equal to null	16n
B,*	2-coordinate string	Length = 0	16n + 4
Z,n	3-coordinate array, short floating point	All three coordinates equal to null	12n
Z,*	3-coordinate string	Length = 0	12n + 4
Y,n	3-coordinate array, long floating point	All three coordinates equal to null	24n
Y,*	3-coordinate string	Length = 0	24n + 4
D	Date and time	Space character filled	20
X	Null field	-	-
K	Triplet id	Type byte = 0	1-13

NOTE: For data types Y and Z, if the elevation (Z) field is not populated due to source restrictions, it will be filled with the appropriate NULL value.

5.4.5 Naming conventions. The following define the naming conventions for VPF file and column names. (All examples shown in this document are in capital letters.) (See also the VPF reserved names in TABLEs 13, 14, and 15.)

a. All naming will use ISO 646 (ASCII) characters.

b. All file names and all references to file names shall be lowercase. This includes file references within table headers, attribute values, and indices. All references to database, library, coverage, and feature class names shall be in lowercase where they occur. For example, feature class names in FCS and FCA tables will be lowercase. For file names, the first character must be an alpha character from a to z. The remaining 7 characters can be alphanumeric, including the underscore ('_') character.

A file name may have a trailing period with a 3-character suffix. The suffix may contain only characters from a to z, except that x is not allowed.

Any table with variable-length records will maintain a variable-length index file with the same file name as its associated table except that the last character will be x (with one exception; see paragraph 5.3.1.2).

c. All names are to be in lowercase.

d. Directory names (names for libraries, databases, and coverages) are restricted to the same rules as for file names, except that there will be no suffix.

e. Column names follow the same restrictions as file names, but they can be 16 characters in length. The dollar sign ('$'), pound sign ('#'), dash ('-'), period ('.'), and slash ('/') are allowable characters.

The column name ("ID") is reserved and must be used to identify each table record.

f. If a column is defined with a triplet id, the fields within the triplet id will be named as:

Field one	ID	The internal tile primitive id
Field two	TILE_ID	The tile reference id
Field three	EXT_ID	The external tile primitive id

The (\) will be used as a separator between the column name and the triplet id field. Thus, when referring to the internal primitive id within a triplet id column (LEFT_FACE) the column name will be named "LEFT_FACE\ID".

5.4.6 Triplet id field type. As discussed in cross-tile keys (section 5.2.2.3.4), a triplet id can be used to reference primitives from multiple tiles in a tiled coverage. This field type replaces the integer foreign key used in untiled coverages. The first component of a triplet is an 8-bit type byte. The type byte is broken down into four 2-bit pieces; each of these 2-bit pieces describes the length of a succeeding field. TABLE 63 lists the possible values for these 2-bit field descriptors. Only the first three fields are currently being used. The fourth field is reserved. FIGURE 20 is an example of the triplet id field.

TABLE 63. Type byte definitions.

Bit Count	Number Bits in Field
0	0
1	8
2	16
3	32

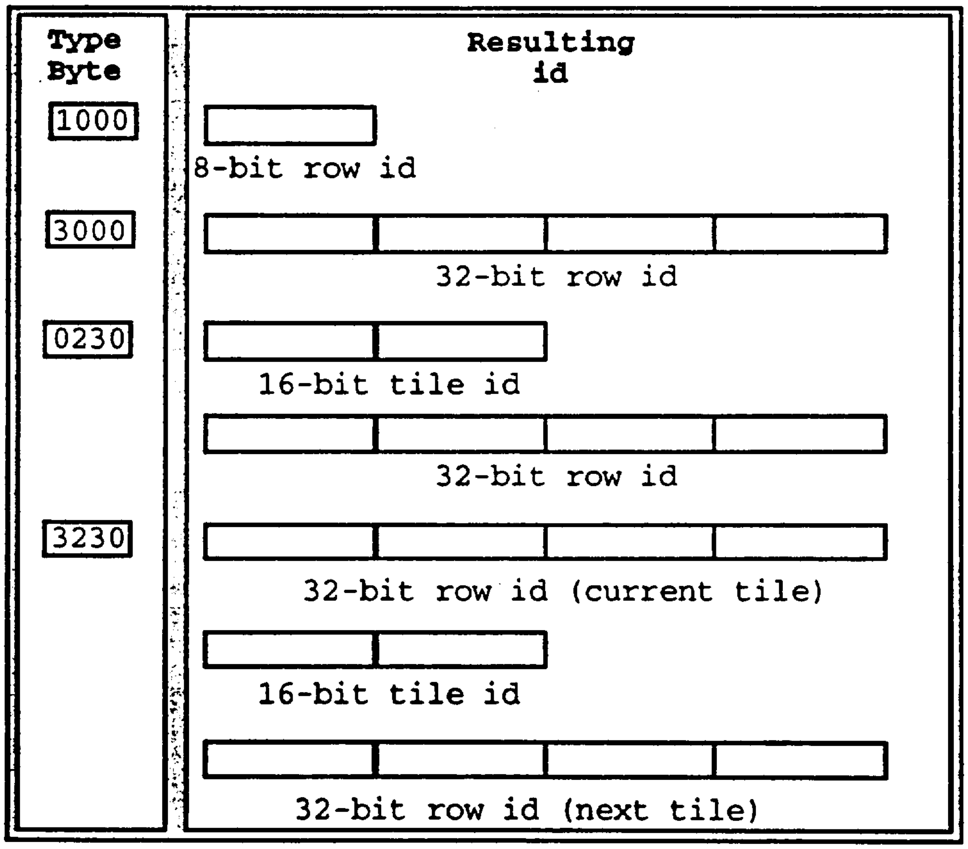
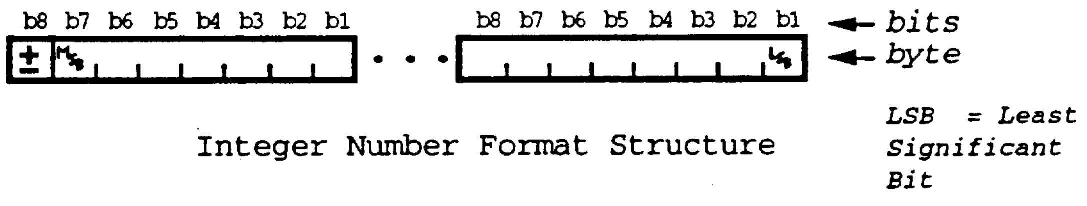


FIGURE 20. Examples of the triplet id.

The first field (referred to as ID) generally is used to store a primitive id without a tile id predicate. The tile reference id, the second field (TILE_ID), and the external primitive id, the third field (EXT_ID), together store an augmented primitive id for cross-tile topology.

5.5 Data syntax requirements. VPF requires the use of numeric, textual, coordinate, and date syntax items. These items comprise the lowest level of the VPF design. In order to utilize these items, a number of basic data types are required. These are integer or real numbers, strings of text, and coordinate data types. The coding of these data types is defined in terms of a number of international standards. VPF products may have a byte order specified in the product specification and the table header. Five categories of data syntax items are required in VPF. These are integer numbers, real numbers, text strings, coordinates, and date.

5.5.1 Integer numbers. The two fixed-length integer data fields are 16 bits and 32 bits; hereafter they are termed "short" and "long" precision, respectively. Integers are binary encoded using the 2's complement scheme. For integer number formats, the null value consists of the sign bit set to one and all trailing bits set to zero. Different length integer numbers may be required in different situations. For example, the number 32,000 can be handled in 2 bytes of data, whereas the number 2,147,483,647 will fill 4 bytes of data. The general structure and several examples of the integer number format can be found in FIGURE 21.


b8 b7 b6 b5 b4 b3 b2 b1
±
b8 b7 b6 b5 b4 b3 b2 b1
bits
byte
Integer Number Format Structure
LSB = Least Significant Bit
MSB = Most Significant Bit

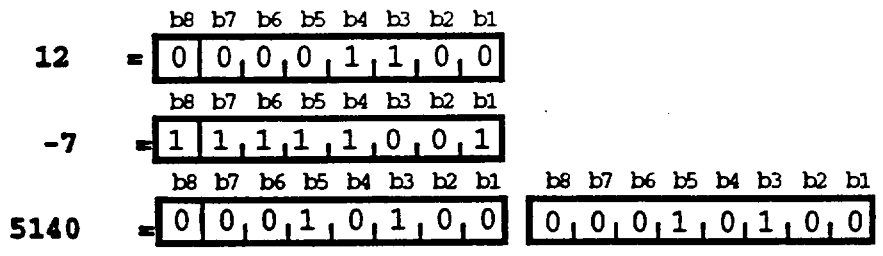


FIGURE 21. Integer number syntax.

5.5.2 Real numbers. Real numbers are needed to carry parametric information. VPF uses the IEEE floating-point real number format (ANSI/IEEE 754) in both 32-bit (short) and 64-bit (long) form. Numbers in the single and double formats are composed of the following three fields:

s	1-bit field for sign
e	Biased exponent field (equals exponent E plus bias)
f	Fraction field (mantissa)

a. Range. The range of the unbiased exponent includes every integer value between E_{min} and E_{max}, inclusive, and also two other reserved values, $E_{min} - 1$ and $E_{max} + 1$, to encode certain special states as described below. Figure 22 illustrates real number syntax.

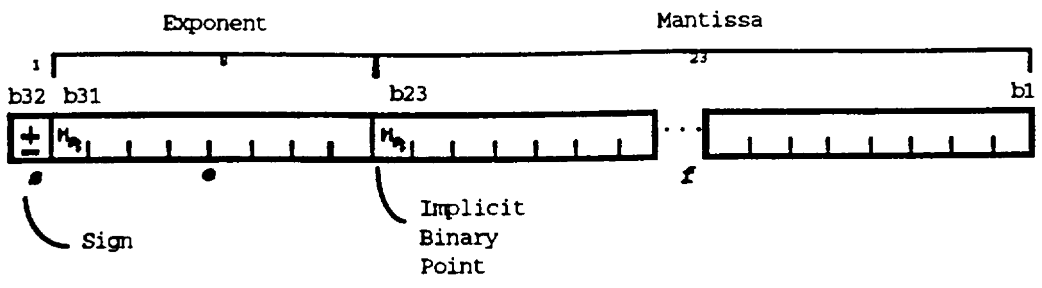
b. 32-bit format. For a 32-bit single format number, the value v is inferred from its constituents thus:

(1) If $e = 255$ and $f \neq 0$, then v is NaN,
(2) If $e = 255$ and $f = 0$, then $v = (-1)^s \infty$,
(3) If $0 < e < 255$, then $v = (-1)^s 2^{e-127} (1 \bullet f)$,
(4) If $e = 0$ and $f \neq 0$, then $v = (-1)^s 2^{e-127} (0 \bullet f)$ (denormalized numbers),
(5) If $e = 0$ and $f = 0$, then $v = (-1)^s 0$ (zero).

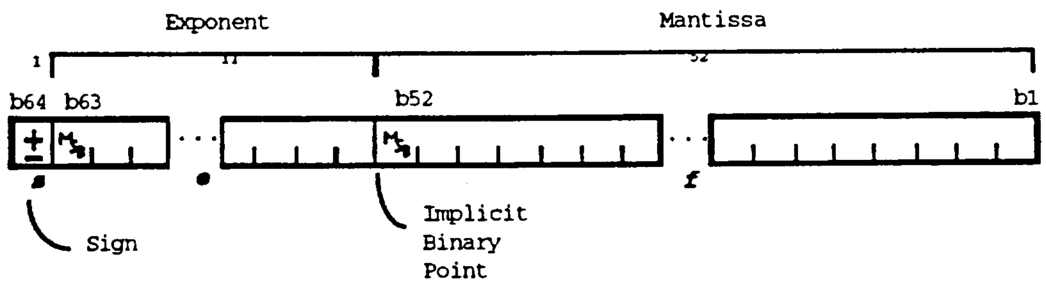
c. 64-bit format. For a 64-bit double format number the value v is inferred from its constituents, thus:

(1) If $e = 2047$ and $f \neq 0$, then v is NaN,
(2) If $e = 2047$ and $f = 0$, then $v = (-1)^s \infty$,
(3) If $0 < e < 2047$, then $v = (-1)^s 2^{e-1023} (1 \bullet f)$,
(4) If $e = 0$ and $f \neq 0$, then $v = (-1)^s 2^{e-1023} (0 \bullet f)$ (denormalized numbers),
(5) If $e = 0$ and $f = 0$, then $v = (-1)^s 0$ (zero).

Note: the "•" in equations (3) and (4) above corresponds to a decimal point.



IEEE Short Real Number Format Structure (32 bit)



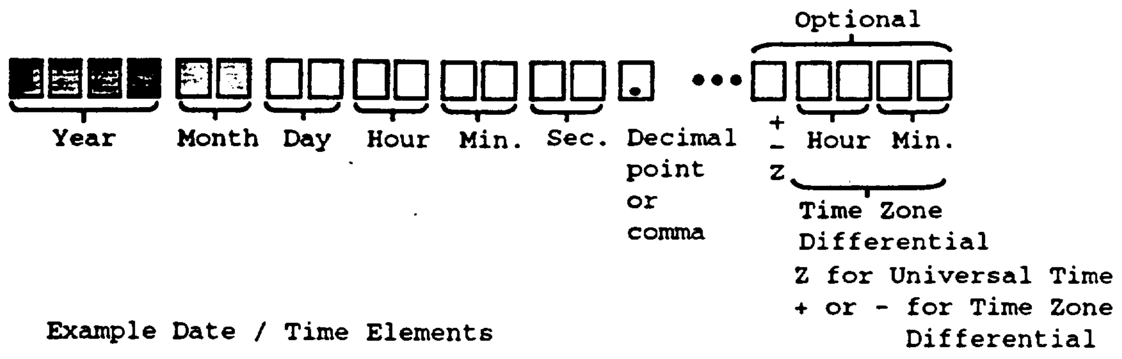
IEEE Long Real Number Format Structure (64 bit)

Where :

- The sign bit is 0 for positive and 1 for negative.
- The exponent is 8 bits long for short real numbers and 11 bits for long real numbers. The exponent is biased by 81 hexadecimal for short real numbers and 401 for long.
- The remaining bits are the mantissa. Since the first significant bit is known to be set (since the mantissa is normalized), it is not stored. The length is 23 bits for short real numbers and 52 bits for long real numbers.

FIGURE 22. Real number syntax.

5.5.3 Date and time syntax. The generalized time data type consists of a string of international reference version (IRV) characters where the calendar date (as specified in ISO 8601) consists of a four-digit representation of the year, a two-digit representation of the month, and a two-digit representation of the day. This is followed by the time of day (as specified in ISO 8601) consisting of a string of digits containing a two-digit representation of the hour (based on the 24-hour clock), a two-digit representation of the minute, and a two-digit representation of the second, followed by a decimal point (or decimal comma) and an arbitrary number of digits of fractions of a second. This may be followed by the letter Z to represent coordinated Universal Time rather than local time, or be followed by a time differential from UT in accordance with ISO 8601 (see FIGURE 23). In a fixed data field usage, date and time elements will be 20 bytes long,



Example Date / Time Elements

▪ ▪	• No date/time given, 20 space characters
1992	• Year is only value given, no time. Pad with space characters.
199210	• Year/month given, no time. Pad with space characters
19870205160627.	• Local time 6 minutes, 27 seconds after 4 pm on 5 February 1987
19870205210627.Z	• As above but Universal Time (Greenwich)
19870205160627.-0500	• Local Time as above. The local time is 5 hour time behind UTC

FIGURE 23. Date and time syntax.

allowing for the specification of date and time with time zone differentials (or optionally fractional seconds). Unfilled digits will be filled with space characters. A null date time specification will consist of a string of space characters.

5.5.4 Text syntax. Text syntax is described in DIGEST Part 3 sections 5.1.4 - 5.2.3. Use the text syntax described for DIGEST Annex C, Vector Relational Tables.

a. Text strings. Textual information can be either variable length or fixed length. The null state of a variable-length text string is of zero length. The null state of a fixed-length text string requires that a specific code be selected. The character string "N/A" should be used, padded if necessary. If the length is one or two, "-" or "--" should be used instead. The CO (control set code table) character SP (code table position 2/0 in FIGURE 24) should be used as the "space" or "blank" character, and as the padding character. The character code NUL (code table position 0/0) and a number of other CO control characters may have special meaning on some computer systems and should not appear in any text strings. A NUL or a SUB (^Z) in a file is an end of file mark on some computers. Two types of text strings are supported in VPF:

1. Basic text string. These strings make use of characters only from the IRV (ASCII) primary code table and the subset of the CO table identified above.

2. General text strings. These strings are composed of characters from any of the ISO registered code tables. Code extension (ISO 2022) is only required to handle written languages that are not based on the Latin alphabet. For languages (like English, French, German, or Spanish) that use the Latin alphabet, the IRV (ASCII) and the supplementary code table together with the identified subset of the CO set will be used.

b. Code tables. All text is coded in terms of character sets. Particular character codes are identified by a code table arranged into rows and columns in which 94 character codes are assigned. A number of different character code tables are in use internationally and these code tables are registered with the International Standards Organization under ISO 2375. The basic international code table is the IRV alphabet (see FIGURE 24).

The alphabetic code table is termed the graphic or "G" set. Another specialized code table, the control or "CO" set, is also defined. Some of the CO control characters are reserved for specialized use, such as transmission control in an asynchronous communications system or application level delimiting. VPF requires only the format effector CO characters, such as carriage return (CR) and line feed (LF), and the code extension characters escape (ESC), shift in (SI), and shift out (SO). The code extension characters allow extension to other alphabets as described below. Other CO characters are not used and have a null meaning. The IRV (ASCII) code table caters largely to the needs of the English language. For other Latin languages in which accented letters are used extensively, the ISO has recommended a supplementary character set for coded characters in text communication (ISO 6937). First, a nonspacing accent character is selected from the supplementary character set; then the accented character is selected.

´ + e = é

				column								
				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	row	0	1	2	3	4	5	6	7
0	0	0	0	0	[illegible]	[illegible]	SP	0	@	P	`	p
0	0	0	1	1	[illegible]	[illegible]	!	1	A	Q	a	q
0	0	1	0	2	[illegible]	[illegible]	"	2	B	R	b	r
0	0	1	1	3	[illegible]	[illegible]	#	3	C	S	c	s
0	1	0	0	4	[illegible]	[illegible]	$	4	D	T	d	t
0	1	0	1	5	[illegible]	[illegible]	%	5	E	U	e	u
0	1	1	0	6	[illegible]	[illegible]	&	6	F	V	f	v
0	1	1	1	7	[illegible]	[illegible]	'	7	G	W	g	w
1	0	0	0	8	BS	[illegible]	(	8	H	X	h	x
1	0	0	1	9	HT	[illegible]	)	9	I	Y	i	y
1	0	1	0	10	LF	[illegible]	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	[illegible]	,	<	L	\	l	|
1	1	0	1	13	CR	[illegible]	-	=	M	]	m	}
1	1	1	0	14	SO	[illegible]	.	>	N	^	n	~
1	1	1	1	15	SI	[illegible]	/	?	O	_	o	[illegible]

FIGURE 24. Latin alphabet primary code table (ASCII).

This supplementary code table (see FIGURE 25) may be used with IRV (ASCII) or other national variants of IRV. In addition, the repertoire of all accented characters and diacritical marks (see FIGURE 26) covers all Latin alphabet-based languages as represented by ISO 6937.

For languages based on other alphabets (such as Greek, Cyrillic, Chinese Hanzi, and Japanese Kanji or Katakana), independent code tables may be defined. These code tables are registered with the ISO and are assigned a final character code for use with a designated escape sequence. By use of these escape sequences, the current "in-use" code table may be switched.

The code table switching mechanism is specified by ISO 2022. The "in-use" code space is organized into rows and columns and divided into two areas, the "in-use" C area and the G area. The character ESC (escape) (position 1/11) is used as an introducer to sequences of codes which determine which code table is in use.

				column								
				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
row					0	1	2	3	4	5	6	7
b4	b3	b2	b1									
0	0	0	0	0	▨	▨	NB SP	°	▨	―	Ω	ĸ
0	0	0	1	1	▨	▨	¡	±	`	‘	Æ	æ
0	0	1	0	2	▨	▨	¢	²	´	®	Đ	đ
0	0	1	1	3	▨	▨	£	³	^	©	ª	ð
0	1	0	0	4	▨	▨	$	×	~	™	Ħ	ħ
0	1	0	1	5	▨	▨	¥	µ	¯	♪	▨	ı
0	1	1	0	6	▨	▨	#	¶	˘	¬	Ĳ	ĳ
0	1	1	1	7	▨	▨	§	·	˙	¦	Ŀ	ŀ
1	0	0	0	8	▨	▨	¤	÷	¨	▨	Ł	ł
1	0	0	1	9	▨	▨	‘	’	▨	▨	Ø	ø
1	0	1	0	10	▨	▨	“	”	˚	▨	Œ	œ
1	0	1	1	11	▨	▨	«	»	¸	▨	º	ß
1	1	0	0	12	▨	▨	←	¼	▨	⅛	Þ	þ
1	1	0	1	13	▨	▨	↑	½	˝	⅜	Ŧ	ŧ
1	1	1	0	14	▨	▨	→	¾	˛	⅝	Ŋ	ŋ
1	1	1	1	15	▨	▨	↓	¿	ˇ	⅞	ŉ	SHY

FIGURE 25. Latin alphabet supplementary code table of accents, diacritical marks, and special characters.

a. International alphabets. The following is a list of the most common alternate alphabets from the international registry, together with their final character for the escape sequence used to designate.

IRV (Latin) alphabet	-	4/0
UK variant of IRV	-	4/1
ASCII variant of IRV	-	4/2
Other set for use with variants of IRV	-	6/12
Katakana alphabet (Japanese)	-	4/9
Greek alphabet	-	6/10
Cyrillic alphabet	-	4/14
Extended Cyrillic alphabet	-	5/1
African languages alphabet	-	4/13
Arabic alphabet	-	6/11

Basic Letter	Acute Accent	Grave Accent	Circumflex Accent	Diaeresis or Umalut	Tilde	Caron	Breve	Double Accute Accent	Ring	Dot	Macron	Umalut Mark	Cedilla	Ogonek
aA	áÁ	àÀ	âÂ	äÄ	ãÃ		ăĂ		åÅ		āĀ	äÄ		ąĄ
bB														
cC	ćĆ		ĉĈ			čČ				ċĊ			çÇ	
dD						ďĎ								
eE	éÉ	èÈ	êÊ	ëË		ěĚ				ėĖ	ēĒ	ëË		ęĘ
fF														
gG	ǵ		ĝĜ				ğĞ			ġĠ			Ģ	
hH			ĥĤ											
iI	íÍ	ìÌ	îÎ	ïÏ	ĩĨ					i	īĪ			įĮ
jJ			ĵĴ											
kK													ķĶ	
lL	ĺĹ					ľĽ							ļĻ	
mM														
nN	ńŃ				ñÑ	ňŇ							ņŅ	
oO	óÓ	òÒ	ôÔ	öÖ	õÕ			őŐ			ōŌ	öÖ		
pP														
qQ														
rR	ŕŔ					řŘ							ŗŖ	
sS	śŚ		ŝŜ			šŠ							şŞ	
tT						ťŤ							ţŢ	
uU	úÚ	ùÙ	ûÛ	üÜ	ũŨ		ŭŬ	űŰ	ůŮ		ūŪ	üÜ		ųŲ
vV														
wW			ŵŴ											
xX														
yY	ýÝ		ŷŶ	ÿŸ								ÿŸ		
zZ	źŹ					žŽ				żŻ				

FIGURE 26. Usage of accents and diacritical marks.

b. Chinese and Japanese alphabets. The Chinese and Japanese iconographic alphabets may also be designated. These character sets are special in that they require two consecutive bytes to index into over 8,000 entries. The designation sequences are given below:

Chinese Hanzi - ESC 2/4 4/1

Japanese Kanji - ESC 2/4 4/2

5.5.5 Coordinate syntax. A coordinate specifies a position in the Cartesian unit coordinate space as a vectorial displacement from the origin of the coordinate space. A coordinate parameter value takes the form of an short or long floating point value.

5.5.6 Coordinate strings. Two types of coordinate strings are defined for use in VPF. These consist of coordinate tuples (pairs or triplets). All coordinate strings are constructed out of the number and coordinate formats defined in the previous subsections. A coordinate string consists of a sequence of coordinate parameter values corresponding to coordinate pairs.

6. NOTES

(This section contains information of a general or explanatory nature that may be helpful, but is not mandatory.)

6.1 Intended use. This standard is designed to define the methods and provide guidance for creating and using digital geographic databases in vector product format.

6.2 Acquisition requirements. When this standard is used in acquisition, the applicable issue of DODISS must be cited in the solicitation (see 2.1.1 and 2.2).

6.3 Supersession. These standard supersedes Military Standard for Vector Product Format, MIL-STD-600006, 13 April 1992.

6.4 Subject term (key word listing).

Database
Geographic information
Georelational data
Geospatial features
Metadata
Spatial data

6.5 Changes from previous issue. Marginal notations are not used in this revision to identify changes with respect to the previous issue due to the extent of the changes.

APPENDIX A

INTRODUCTION TO THE VPF DATA MODEL

A.1. GENERAL

A.1.1. Scope. This appendix provides information, discussion, and examples concerning the VPF data model. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

A.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

A.3. DEFINITIONS

A.3.1. Definitions used in this appendix. For purposes of this appendix, the definitions in section 3 of the main document shall apply.

A.4. GENERAL INFORMATION

A.4.1. Introduction. VPF is a general, user-oriented data format for representing large spatially referenced (geographic) databases. VPF is designed to be used directly; that is, software can access the data without time-consuming conversion processing. VPF is designed to be compatible with a wide variety of users, applications, and products.

To achieve its generality and user orientation, VPF uses a georelational data model that provides a flexible but powerful organizational structure for any digital geographic database in vector format. VPF defines the format of data objects, and the georelational data model supporting VPF provides a data organization within which software can manipulate the VPF data objects.

The following paragraphs discuss in general the data model that serves as the basis of VPF. Section A.4.2 discusses the basic concepts that form the foundation for all geographic data models. Section A.4.3 describes the relational model, while section A.4.4 describes the planar topology model.

A.4.2. Data model concepts. A model is a fundamental description of a system that accounts for all known properties of that system. The system is a view of geographic reality, or information tied to specific locations in coordinate space. Since this particular model is stored in a computer, it is called a data model. A data model provides the most abstract representation of a system; it describes a collection of entities, including their relationships and semantics. The purpose of a data model is to define and capture a view of reality in a consistent and uniform manner. It provides a framework to visualize the structure and behavior of these entities in a system.

A model of a database requires three components: the definition of the data objects, data operations, and the rules of data integrity. These components are defined in the following three subparagraphs. Objects identify how the user perceives the data and its structure. The operations define how the user may manipulate the objects. Integrity rules bind the objects and operations, and establish a well-defined behavior that provides accurate information in a predictable manner. The fourth sub-paragraph deals with the purpose and functionality behind a database.

A.4.2.1. Data objects. Data objects identify how the user perceives the data and its structure. These objects also define the most primitive partitions of the data model architecture. For instance, an international database may contain a wide variety of objects concerning nations. The relevant objects could be areas, civil divisions, populations, resources, products, and water bodies. Relationships can be defined for these objects where populations and resources are grouped by civil divisions.

A.4.2.2. Data operations. Data operations define how a user may manipulate the objects; where, for instance, an object's attributes may be displayed, or new attributes could be defined, or, perhaps, new objects created. In most cases, an algebra is defined that accurately manipulates the data objects and binds the scope of operations within the data model. Classical database operations include retrieval, creation, deletion, and modification. More specific operations are defined for applications using the database.

A.4.2.3. Data rules. The rules of data integrity constrain the operations on objects in order to preserve overall stability. The goal is to prevent operations that yield corrupt, incorrect, or ambiguous results. Integrity rules constrain the set of valid states of databases that conform to the data model. These rules define the accuracy of the database.

A.4.2.4. Database purpose. One function of a database is to provide centralized control of operational data that is vital to an organization. A data model attempts to closely mesh the data objects, operations, and integrity rules into a cohesive system with optimal performance. The advantages of centralized database control are well established, and the use of a data model allows a database to provide the following functionality to an organization.

a. Consistency. The database will provide access to data in a formal manner. This will establish a consistent view of the data, enabling efficient data exchange.

b. Simplicity. A basic objective is to provide an intuitive, straightforward, and understandable interaction between the user and the data.

c. Nonredundancy. Duplication of data will be avoided wherever possible, especially when repetition provides little additional information.

d. Multiple applications. The data model needs to support multiple end user applications because user views of the database will be different.

e. Flexibility. A critical requirement of an effective database is the ability to accept new data. A database needs to have the dynamic flexibility to grow with the needs and requirements of its users.

f. Integrity. The integrity rules should be defined in a consistent manner for all data objects and operations. The use of well-defined rules prevents operations that lead to data corruption and misinformation.

In summary, a data model provides a powerful approach to achieving optimum centralized control of critical data within a system. Using a variety of integrity rules and established operations upon defined data objects, the database provides users with the necessary tools for extracting data in support of many applications.

A.4.3. Relational data model concepts. The relational data model provides a powerful architecture for database design because of its ability to handle a wide variety of data and applications.

A.4.3.1. Relational data objects. The relational model uses simple tabular data structures to portray the data in a natural, well-defined manner. These data structures contain columns and rows, where columns define attributes, the values for which are taken from a range of data defined across a given domain. The content of the rows represents the actual data entities. The strength of the relational model is its ad hoc ability to establish meaningful data, transforming data into information as a function of user perspective. For instance, a relational table "ROADS" can be defined as having three columns: name, class, and structure. The rows that compose the ROADS table would contain distinct information about each road in each field in the database. The six following properties distinguish relational tables from nonrelational data objects:

a. Entries in columns must be single-valued; a field may not contain a list of attributes.

b. Entries within a column must be of the same data type.

c. Each row must be unique, and duplicate rows are not permitted.

d. Columns may appear in any order.

e. Rows may also appear in any order.

f. Each column must have a unique name.

A.4.3.2. <u>Relational data operations</u>. The relational model supports eight set operations: select, project, product, join, union, intersection, difference, and division. Since the relational model is founded on set theory, the operations themselves are based on fundamental mathematical principles. These operations allow data objects to be manipulated and created in a specific manner, producing stable results.

A.4.3.3. <u>Relational data rules</u>. The integrity rules constrain operations performed on objects in order to preserve stability. The goal is to prevent operations that yield corrupt, incorrect, or ambiguous results. The entity integrity rule requires the entry of a null value in columns in relational tables for which the value is always known or understood. The referential integrity rule ensures that a foreign key referenced into another table stays within recognizable bounds. For example, a foreign key is not permitted that references a record number of 500 in a table that only contains 300 records. Additional, more subtle domain rules can also be defined to constrain entries in and operations on the database.

A.4.4. <u>Plane topology model concepts</u>. Conceptually, plane topology can be defined as a planar graph, where geographic reality is decomposed into a finite set of 0 cells (nodes), 1 cells (edges), and 2 cells (faces). This terminology is defined by an algebraic topology that establishes rules for decomposing continuous three-dimensional objects into representations of finite models. Once this topologic mapping has been performed, a system can be modeled in a way that permits more complex relationships between objects to be established.

The purpose of topology is to capture and retain knowledge concerning a cell's spatial and thematic relationships with its neighboring cells. For a topologic model to be valid, these relationships must remain constant regardless of changes in scale, shape, or size. With topology embedded in a data model, very useful relations can be established, such as adjacency and connectivity. Topologic and geometric relationships (such as size, angle, and shape) provide powerful resources that allow geographic reality to be fully modeled.

A.4.1.1. Topologic objects. Plane topology extends graphic models of nodes and edges to the development of a more powerful and expressive model that contains spatial relationships. In addition to metric capabilities (distance, shape, or size), topology determines spatial neighbor relations. By defining more rigorous and complex relationships in the data model, the true properties of a system can be more effectively represented. Various mathematical models are available. In addition, simpler models can be used to create more complex ones. The most simple is the graphic model. More complex and useful surface-based models are built upon the graphic, using topology to capture additional analytical information.

A graphic model represents geometric information based on node and edge primitives. This model provides the base for other models that define more complex relationships. The node primitive is composed of a location in an established coordinate space. Most representations are two dimensional (x,y) or three dimensional (x,y,z). An edge is composed of a minimum of two nodes, with more details concerning linear or spline interpolation between the end points.

Because of the complexity of geographic information and the limitations of a graphic model, plane topology provides a better model for defining relationships. The use of a planar topology model (based on the two-dimensional manifold), for instance, maps more concisely in the two-dimensional space that current computing systems can manage. VPF provides four distinct levels of topology: full planar topology (level 3); a linear planar graph (level 2); a nonplanar linear graph (level 1); and no topology defined, indicating a geometric model (level 0). VPF uses the notion of topology as a constraint to enforce integrity rules upon the feature definitions. As the entities require fewer topological relationships, the rules can be relaxed. For instance, if linear features in a transportation network are being modeled, then the requirement of full planar topology may be relaxed because it is not necessary.

Plane topology defines relations between cells without modifying the underlying geometry. The concept of a cell can be visually retained in graphic models, but only allows one view of the data. Topology establishes a framework that provides more information for analysis. For instance, in figure 27, the edges 1, 2, 3, 4, and 5 are grouped together into face 'a.' Another face known as 'b,' including the edges 5 through 12, can be defined. Topology defines relationships on each 0-, 1-, and 2-cell in a model. Face 'b' is defined to be "left-of" edge 5; edge 7 can be defined to

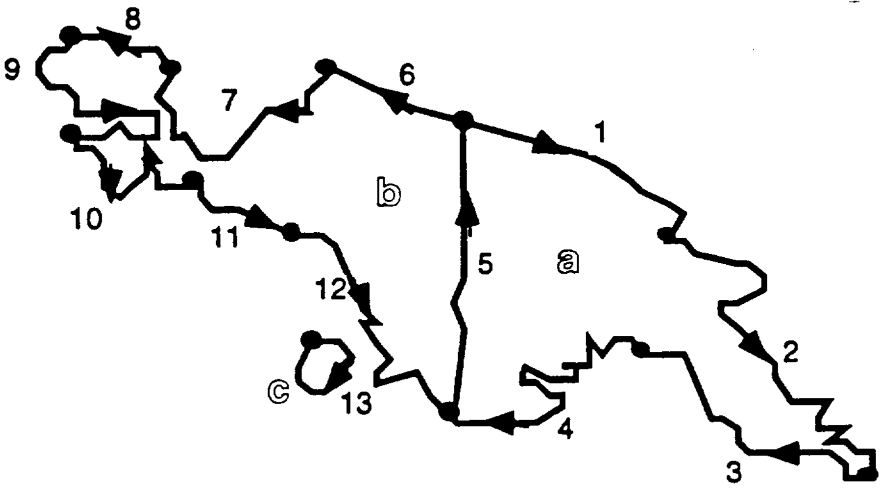


FIGURE 27. The definition of faces.

follow edge 6 in a cycle. This topologic information provides the power to determine orientation, adjacency, and connective relationships between objects.

The concept of topology held by the geometric primitives is carried upward to features and their associated thematic information. An area feature is usually labeled, or contains information pertaining to the enclosed region. For instance, an area can have a category (soil class, surface material type) or a numeric value (population size, number of airports). Topology is used to provide operations and information to distinguish between these thematic objects. Thematic relationships can exist between (geometry, topology, and relational tables) provides a robust database architecture.

VPF adheres to the georelational data model, but only defines the objects and the data structures that compose the objects. The georelational operations and algebra are not part of the standard, but rather are implemented in software. Every VPF object is described in the form of a relational table, composed of columns defining the syntax of each field and rows that contain the actual data.

features without requiring the primitive geometry. For instance, a set of the islands in the Pacific Ocean (Oahu, Maui, Hawaii, Molokai, etc.) can be defined as different features with differing geometric primitives, but they can be related to one another as the Hawaiian Islands.

A.4.1.2. Topological operations. Topological operations are based on the single notion of adjacency—that is, if two objects are next to each other, it is necessary to maintain the adjacent relationship between them. To distinguish the topological aspects from the geometric aspects of geography, we are only concerned with whether two objects, A and B, are adjacent to each other and not with whether A is bigger than B, or one is to the north of the other, or the length of their common boundary.

Many complex topological operations can be derived from adjacency alone. In the georelational data model implemented for VPF, two topological operations are paramount: boundary and coboundary. For example, an edge has a start node and an end node; the nodes are the boundary for the edge. The edge, in turn, is the coboundary of the node. Of course, the coboundary of a node can have more than one edge if many edges meet at a node. Similarly, faces have edges as boundaries. The coboundaries of edges are maintained in the left and right faces.

A.4.1.3. Topological rules. The integrity rules of the topological model are contained in the definition of the objects themselves. A plane model restricts itself to planar geometry, where all entities must lie in the same plane. In addition, all faces must be mutually exclusive and not overlapping. These constraints allow the objects to be defined in context and allow operations to be performed in a consistent manner. While these rules may seem to restrict the system model, they define the data model's domain, taking advantage of the underlying structures. By restricting the faces to be constructed of nonoverlapping regions, powerful set operations can be applied to the objects (such as union, intersection, or join).

A.4.1.4. The VPF georelational data model. VPF uses a combination of the relational and the planar topologic data models to provide a hybrid model for geographic data management, analysis, modeling, and display. The georelational model provides the data structure foundations for a spatial database, and software provides the rules and operators that manipulate topology, geometry, and relational objects in the form of tables. Whenever an operation requires thematic information, the use of relational and topologic table operations are used to supply the result. If the operation is spatially related, geometry and topology together will be used. This triad of categories

APPENDIX B

WINGED-EDGE TOPOLOGY

B.1. GENERAL

B.1.1 Scope. This appendix provides information, discussion, and examples concerning winged-edge topology. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

B.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

B.3. DEFINITIONS

B.3.1 Definitions used in this appendix. For purposes of this appendix, the definitions of section 3 of the main document shall apply.

B.4. GENERAL INFORMATION

B.4.1 Winged-edge topology. Winged-edge topology is an essential part of the VPF data model. The function of winged-edge topology is to provide line network and face topology and also to maintain seamless coverages across a physical partition of tiles. The following sections define the components of winged-edge topology, the algorithm used to traverse the winged-edge network, and cross-tile topology.

B.4.2 Components of a winged edge. Winged-edge topology uses three specific components (columns) on an edge primitive table to provide connectivity between nodes, edges, and faces. Given level 1, 2, or 3 topology, the edge primitive will contain specific columns for each topologic level. As shown by FIGURE 28, there are three topologic constructs: node, edge, and face information on each edge. These constructs are formally defined in section 5.3. A brief summary of the definition is repeated below.

a. Node information. Each edge will contain a start node and an end node column. This topologic information is used to define an edge direction (digitizing direction).

b. Edge information. Right and left edges connect an edge to its neighbor edges (thus the term "winged edge"). The right edge is the first edge connected to the end node that is encountered when cycling around the node in a counterclockwise direction. The left edge is the first edge connected to the start node that is encountered when cycling around the node in a counterclockwise direction.

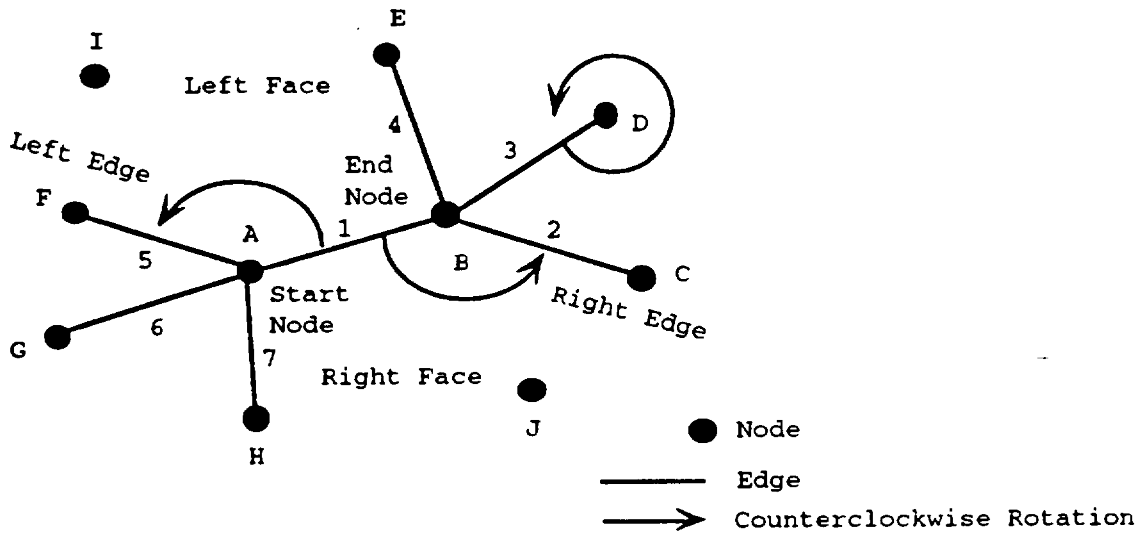
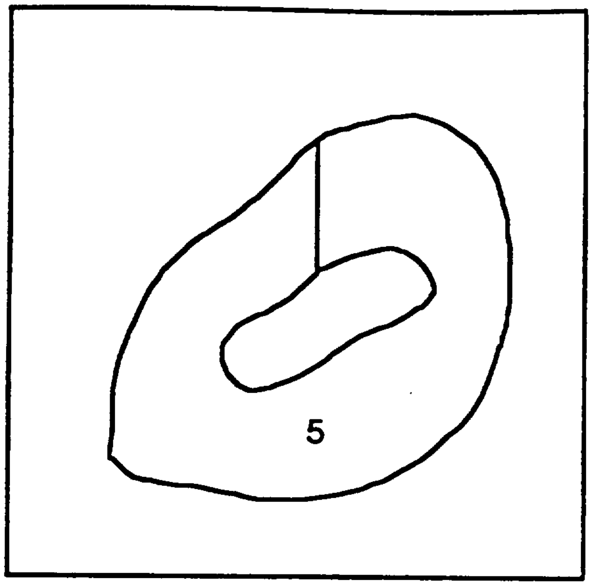


FIGURE 28. Winged-edge components.

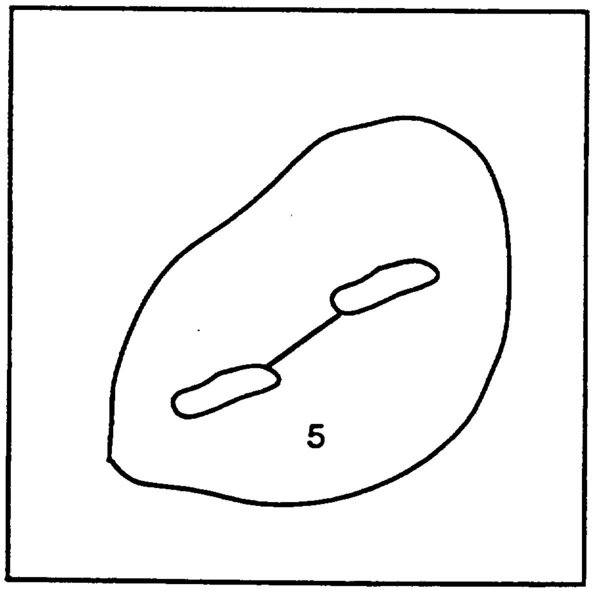
c. Face information. With level 3 topology specified, each edge will contain a left and right face. Left and right face are determined solely by the edge direction. This information allows an edge to know its neighboring faces.

B.4.2.1 Inner rings. The composition of a face's outer and inner rings are governed by the rules of winged-edge topology. In addition, since edges are never considered inside a face but, rather, borders of faces, floating edges within a face will be treated as inner rings. FIGURES 29, 30 and 31 illustrate some cases of outer and inner rings.



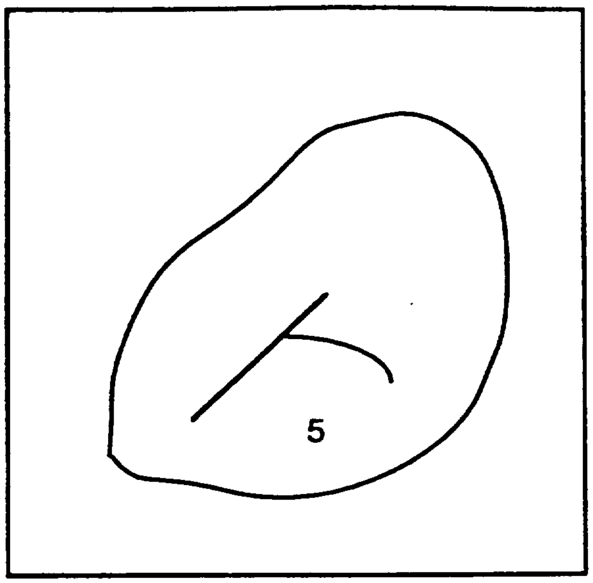
Note: There is no inner ring.

FIGURE 29. Face 5 is represented as a single ring in the ring table.



Note: The two inner faces and their connected edge compose a single inner ring and are identified by a single ring in the ring table.

FIGURE 30. Face 5 is represented as two rings in the ring table.



Note: The second ring in the ring table identifies the floating edge within the face.

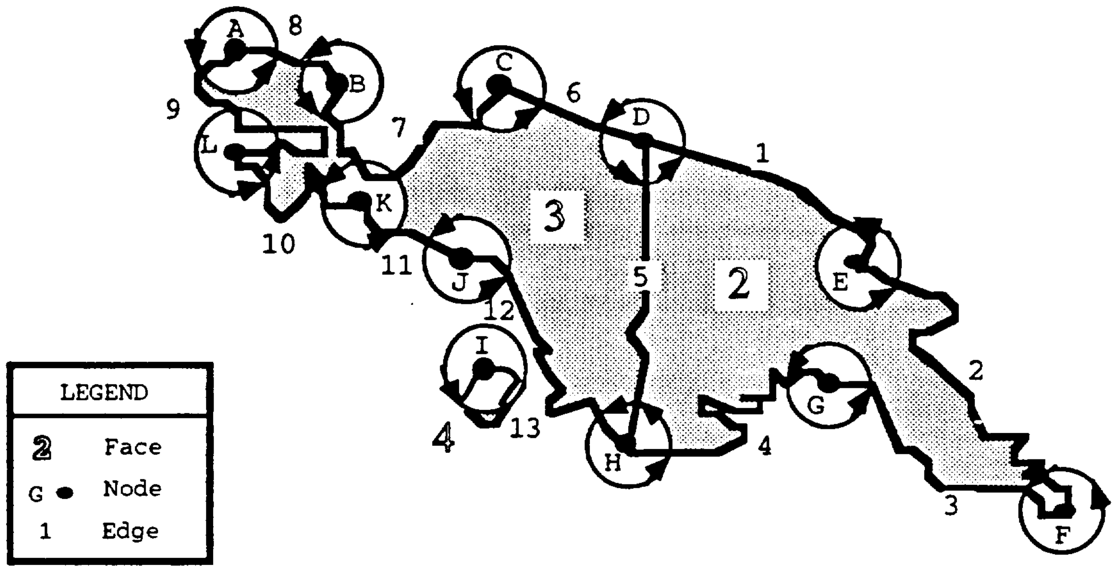
FIGURE 31. Face 5 is represented as two rings in the ring table.

B.4.3 Winged-edge algorithm. Given the definition of a winged edge, every coverage containing faces and edges is created in the same way. With the enforcement of a planar topologic model, a consistent navigation algorithm can be applied.

FIGURE 32 depicts a collection of faces and accompanying edges. To navigate a face, the following algorithm is used:

a. Determine which face to draw. Determining which face to draw would typically be driven by the selection of area features that have the attributes desired; then key through the face and ring tables associated with the area.

   An example of an area feature table consistent with FIGURE 32 is shown in TABLE 62.



ID	START_NODE	END_NODE	RIGHT_FACE	LEFT_FACE	RIGHT_EDGE	LEFT_EDGE	Coordinates
1	D	E	2	1	2	6	
2	E	F	2	1	3	1	Not
3	F	G	2	1	4	2	Shown
4	H	G	1	2	3	5	
5	H	D	2	3	1	12	
6	D	C	1	3	7	5	
7	C	B	1	3	8	6	
8	B	A	1	3	9	7	
9	A	L	1	3	10	8	
10	K	L	3	1	9	11	
11	K	J	1	3	12	10	
12	J	H	1	3	4	11	
13	I	I	4	1	13	13	

FIGURE 32. Winged-edge example, with drawing completely contained within tile 95.

TABLE 64. Sample area feature table for FIGURE 32.

ID	TILE_ID	FAC_ID	<attribute 1>...< >
.	.	.	.
.	.	.	.
.	.	.	.
436	95	3	1 ...
.	.	.	.
.	.	.	.
.	.	.	.

TABLE 64 identifies tile 95 with face 3 as the primitive corresponding to area feature 436. The face table for tile 95 for FIGURE 32 (TABLE 65) yields the following:

TABLE 65. Sample face table for FIGURE 32.

ID	*.AFT_ID	RING_PTR
.	.	.
.	.	.
3	436	7

The ring table for FIGURE 32 (TABLE 66) yields the following:

TABLE 66. Sample ring table for FIGURE 32.

ID	FACE_ID	START_EDGE
.	.	.
.	.	.
.	.	.
7	3	12

b. Now identify the start edge (in this case, 12).

c. Travel to the left edge to trace the left face; the right edge for the right face. Because face 3 is the left face of edge 12, read the left edge record (edge 11). Edge 11 leads to edge 10, edge 10 to edge 9, edge 9 to edge 8, edge 8 to edge 7, edge 7 to edge 6, edge 6 to edge 5, and edge 5 to edge 12.

d. Edge 12 is the start edge, so the cycle is complete.

The same figure can be navigated as a linear network, regardless of the existence of face topology (ring list, left face, right face). Assume that a line feature table exists for this example and that every line feature has an attribute column that is used to determine the correct edge selection in the network. The value of a line attribute will determine the traversal criteria and also determine when it will end. The starting point and endpoint of the traversal are dependent upon the user's application. A network can be traversed with various strategies. The example below illustrates a depth first search. The algorithm is as follows:

a. Locate current edge with user application (edge 12).

b. Read end node of current edge and gather all edges incident at the node.

c. For each of the edges incident at the node, read the attribute value from the associated line feature. Does the attribute have the desired value? If so, continue with step e.

d. Go to step c. Repeat with next edge.

e. Decision. Has the network been completely traversed? If so, exit with complete network. If not, go to step b.

B.4.4 Cross-tile topology. Network navigation using the winged edge can be extended to cross over physical tile partitions, if they exist in the coverage. By using the information in the previous examples, it becomes possible to introduce cross-tile constructs. Assume that FIGURE 32 has been intersected with tile boundaries and that the new coverage in FIGURE 35 has been created (with generalized edges along edge 5 in FIGURE 32).

FIGURE 35 depicts a single face broken into four faces by the intersection of four tiles. The following discussion identifies several different occurrences at the tile boundaries and covers retrieval of the original (untiled) face extent from the tiled faces through winged edge topology.

When creating cross-tile topology, the following rules apply:

a. An edge is always broken when it intersects a tile boundary by placing a connected node at the intersection in both tiles. See FIGURE 34, B and C. All edges terminated by this connected node will have cross-tile topology if an edge exists in the adjoining tiles. See FIGURE 33, A and B.

b. The cross-tile edge will be the first edge in the adjacent tile, counterclockwise from the referencing edge at the node. See FIGURE 33, A and B.

c. All edges which are coincident with a tile boundary (all coordinates for the edge are on the boundary) occur in both tiles (see FIGURE 34, A, D, E and F) and have cross-tile left or right face topology and have cross-tile left and right edge topology. See FIGURE 33, A and B.

d. When a face is broken by a tile boundary, multiple faces are created by closing the face along the tile boundary. See FIGURE 34, A. The edges used to close faces on the boundary are treated as in c. above. See FIGURE 33, B.

e. Connected nodes which occur on tile boundaries, exist in both tiles and reference both an internal and external first edge, if an edge exist. The first edge is selected arbitrarily in both internal and external tiles.

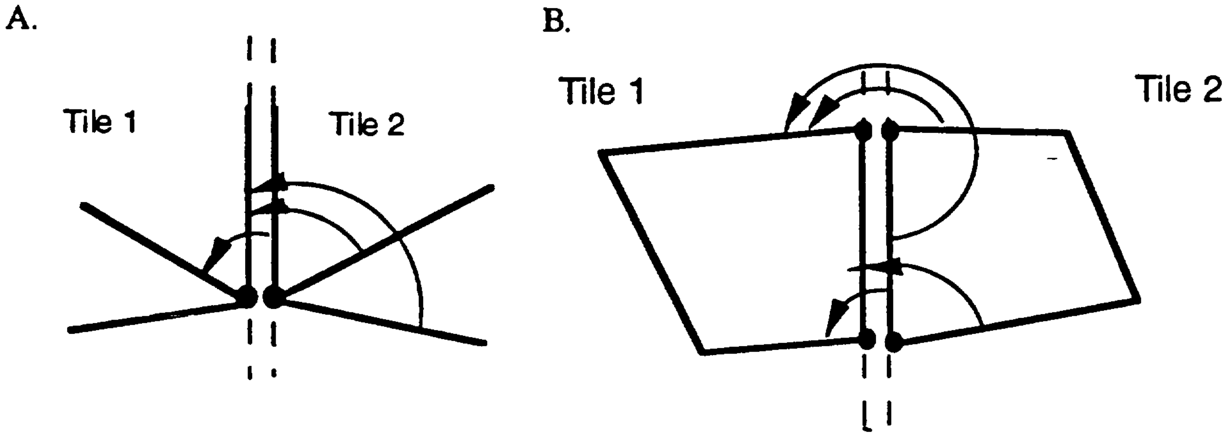


FIGURE 33. Cross-tile edge rules.

The following are additional examples of tile boundary primitive behavior:

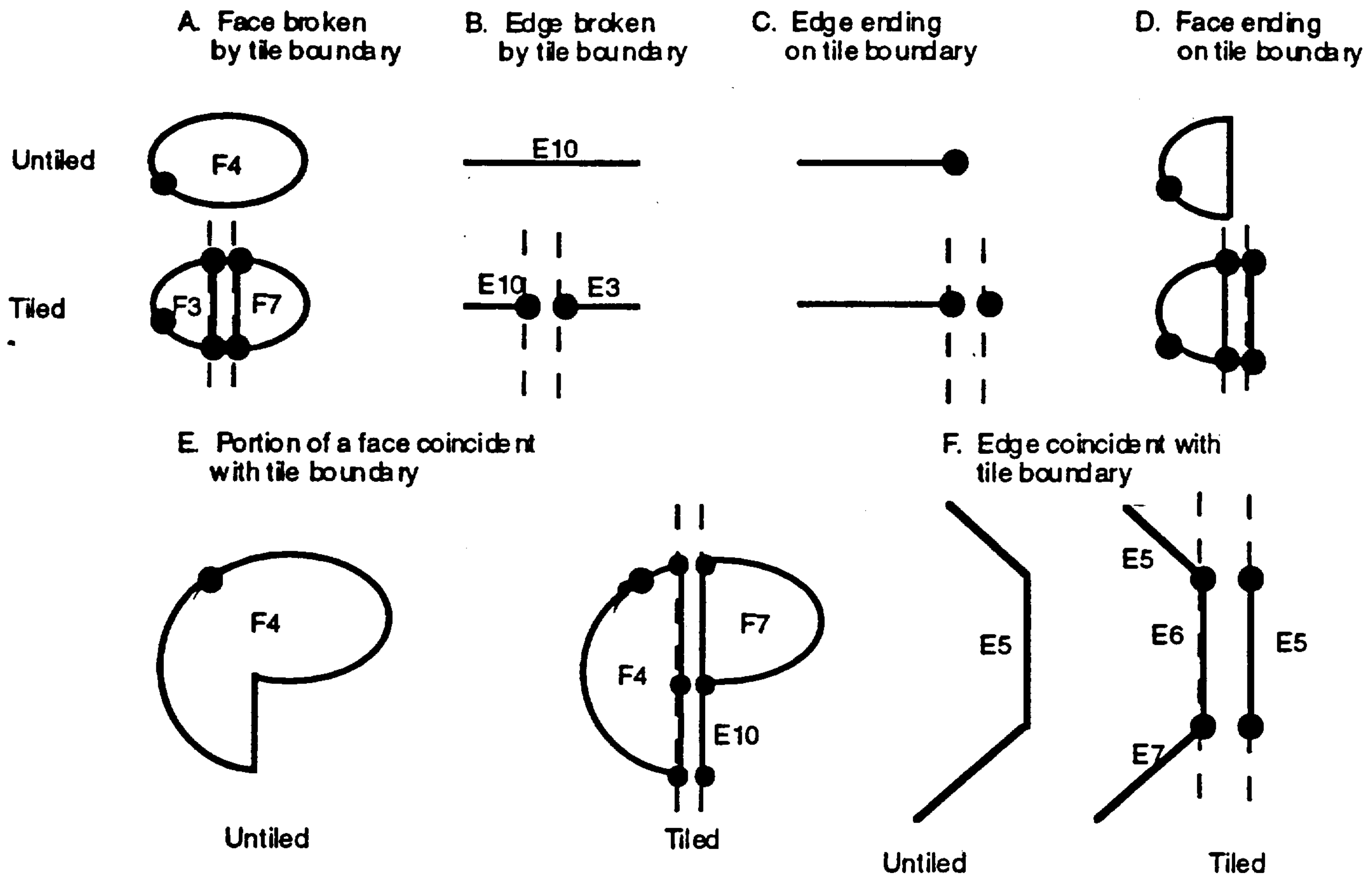
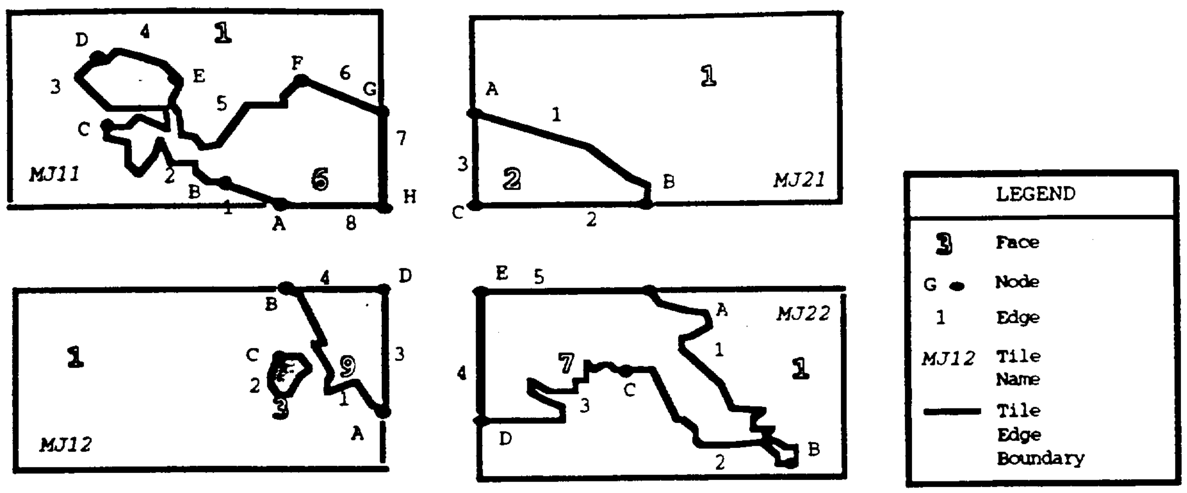

A. Face broken by tile boundary
B. Edge broken by tile boundary
C. Edge ending on tile boundary
D. Face ending on tile boundary
Untiled
Tiled
F4
F3
F7
E10
E10
E3
E. Portion of a face coincident with tile boundary
F. Edge coincident with tile boundary
F4
F4
F7
E10
Untiled
Tiled
E5
E5
E6
E5
E7
Untiled
Tiled

FIGURE 34. Tile boundary primitive behavior



TILE MJ21

ID	Start Node	End Node	Right Face Face,Tile,ExFace	Left Face Face,Tile,ExFace	Right Edge Edge,Tile,ExEdge	Left Edge Edge,Tile,ExEdge	Coordinates
1	A	B	2,-,-	1,-,-	2,MJ22,5	3,MJ11,6	Not Shown
2	B	C	2,-,-	1,MJ22,7	3,MJ11,7	1,MJ22,1	
3	C	A	2,-,-	1,MJ11,6	1,MJ11,6	2,MJ11,8	

TILE MJ22

ID	Start Node	End Node	Right Face Face,Tile,ExFace	Left Face Face,Tile,ExFace	Right Edge Edge,Tile,ExEdge	Left Edge Edge,Tile,ExEdge	Coordinates
1	A	B	7,-,-	1,-,-	2,-,-	5,MJ21,1	Not Shown
2	B	C	7,-,-	1,-,-	3,-,-	1,-,-	
3	D	C	1,-,-	7,-,-	2,-,-	4,MJ12,3	
4	D	E	7,-,-	1,MJ12,9	5,MJ21,2	3,MJ12,1	
5	E	A	7,-,-	1,MJ21,2	1,MJ21,1	4,MJ21,3	

NOTE: Tile names are shown for clarity. The triplet id actually contains the tile id.

FIGURE 35. Cross-tile edge example.

a. Start with tile MJ21, edge 1. Read the left edge. Choose to cross into tile MJ11, edge 6.
b. Chain from edge 6, 5, 4, 3, 2, and 1 within tile MJ11.
c. From edge 1 in MJ11, go across to tile MJ12, edge 1.
d. From edge 1 in MJ12, cross tiles into tile MJ22, edge 3.
e. Chain through edges 3, 2, and 1.
f. From edge 1 in MJ22, cross into tile MJ21, edge 1.
g. When the end of the face cycle is reached, exit.

APPENDIX C

FEATURE CLASS RELATIONS

C.1. GENERAL

C.1.1 Scope. This appendix provides information, discussion, and examples concerning feature class relationships. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

C.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

C.3. DEFINITIONS

C.3.1 Definitions used in this appendix. For purposes of this appendix, the definitions of section 3 of the main document shall apply.

C.4. GENERAL INFORMATION

C.4.1 Overview. One of the most important parts of designing a VPF database is the implementation of a conceptual feature class model into feature class tables. The heart of this implementation concerns the relationships with the feature class tables and their composing primitive tables.

Depending on the design, the relationships between the features and primitives can be one-to-one (1:1), one-to-many (1:N), many-to-one (N:1), or many-to-many (N:M). This appendix is intended to help designers implement their conceptual feature classes with three major implementation constraints: software performance, tiled coverages, and indexing.

C.4.1.1 Software performance. Software performance is a leading consideration in a VPF database design. VPF is designed to support interactive software use. In order for software to respond optimally, the designer must determine the intended use of the database product. If intended for interactive use, the designer should design for optimal performance. If the database is strictly for exchange purposes and not interactive use, a simpler implementation is recommended.

C.4.1.2 Tiled coverages. Tiled coverages require a triplet id column in the feature table or associated join table to define the relationship between features and tiled primitives. Alternatively, the tile and primitive id components of the triplet id can be maintained as separate columns in the feature or join tables. Depending on the feature table type, TABLE 67 lists the column names that shall be used as the join column name.

TABLE 67. Feature table join column definitions.

Area Feature	fac_id
Line Feature	edg_id
Point Feature	end_id
Point Feature	cnd_id
Text Feature	txt_id
Tile ID	tile_id[1]

1. The TILE_ID column is required in tiled coverages when the primitive id column is not defined as data type K (triplet id)

C.4.1.3 Indexing. VPF is designed for interactive use, and the following indexing recommendations apply to help software users achieve these goals.

C.4.1.3.1 Thematic indexes. Implementation of thematic indexes is generally recommended on columns which are likely candidates for thematic queries. For example, thematic indexes on the PRIMITIVE_ID (i.e. FAC_ID, EDG_ID, CND_ID, END_ID, TXT_ID) and TILE_ID columns in feature or join tables can improve software searches based on features located per tile. Note that the VPF Military Std does not support the creation of thematic indexes on columns defined as data type K (triplet id). See TABLE 53 of the VPF Military Standard for a list of allowable data types.

C.4.1.3.2 Spatial indexes. All primitive tables in a VPF database should carry associated spatial indexes. The software performance can be improved significantly.

C.4.1.3.3 Feature Indexes. Feature indexes enhance the retrieval of feature information when given a selected primitive. Implementation of feature indexes is generally recommended when multiple feature classes of the same primitive type exist within a single coverage (i.e. multiple line feature tables within the Transportation coverage; ROADL.LFT and RAILROADL.LFT). Implementation of thematic indexes on the following columns in feature index tables (FIT) is generally recommended to further enhance performance: PRIM_ID, TILE_ID, FC_ID, and FEATURE_ID.

C.4.2 Feature and primitive table relationships. The following subsections provide a number of implementations for feature and primitive table relationships.

Each subsection is based on the five types of relationships (1:1, 1:N, N:1, N:M, and complex). Within each subsection, there are a variety of implementations available, depending on tiling, performance, and index support.

C.4.3 One-to-one relationships. When there is a 1:1 relationship between the feature and primitive tables, the relations are relatively straightforward. Note that for this discussion, a one-to-one relationship between features and primitives means that a feature is composed of one and only one primitive. It does not, however, imply that every primitive is associated with one and only one feature. For example, in a level three coverage with area features, the universe face (face_1) should not be associated with any feature since the outer ring is undefined. Since the Digital Geographic Information Exchange Standard (DIGEST) mandates definition of the feature-to-primitive relation, a PRIMITIVE_ID column (i.e. FAC_ID) is added to the feature table to define the relationship to the appropriate primitive. Only in very unique circumstances (i.e. An untiled, level one or two coverage with one feature class for each primitive type) can the row id in a feature table be used as a foreign key into the row id of the associated primitive table. The reverse relation (primitive-to-feature), although optional for DIGEST compliant data, is recommended for performance reasons.

C.4.3.1 1:1 feature class in an untiled coverage. The untiled 1:1 design (FIGURE 36) is the simplest implementation. Its performance is optimal going from the feature to its primitive. The PRIMITIVE_ID column in the feature table acts as a foreign key, providing a direct link to the associated record in the primitive table. However, the relationship going from the primitive back to the feature is referred to as an indirect link since the PRIMITIVE_ID column of all records in the feature table must be examined sequentially to find a match for a selected primitive. Such indirect links provide poor performance and are generally not recommended.

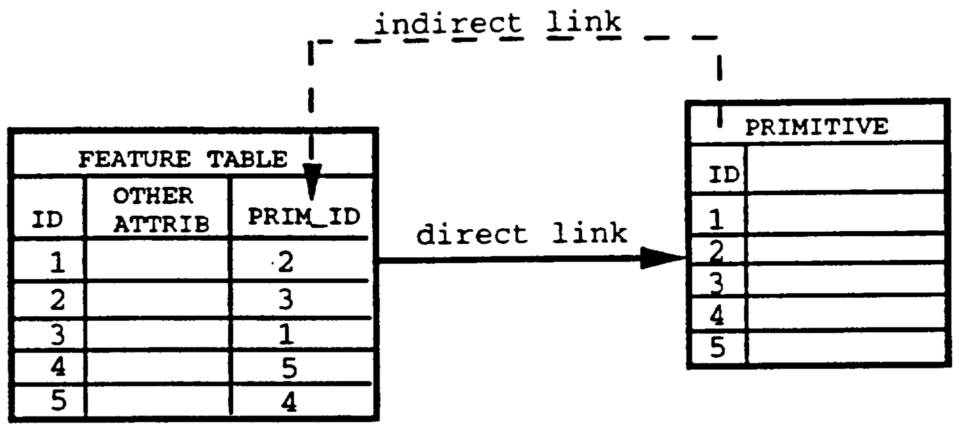


FIGURE 36. Implementation of a 1:1 feature class in an untiled coverage.

- Direct links provide good performance going from the feature to its primitive.

- However, indirect links going from the primitive back to the feature provide poor performance.

C.4.3.2 1:1 feature class in a tiled coverage. The tiled 1:1 design (FIGURE 37) is required to maintain direct relations between one feature and one primitive stored in only one tile. The performance of this implementation is good going from the feature to its primitive. However, performance going from the primitive back to the feature is poor because of the indirect link. For example, the PRIMITIVE_ID and TILE_ID columns of all records in the feature table must be examined to find all matches for a selected primitive.

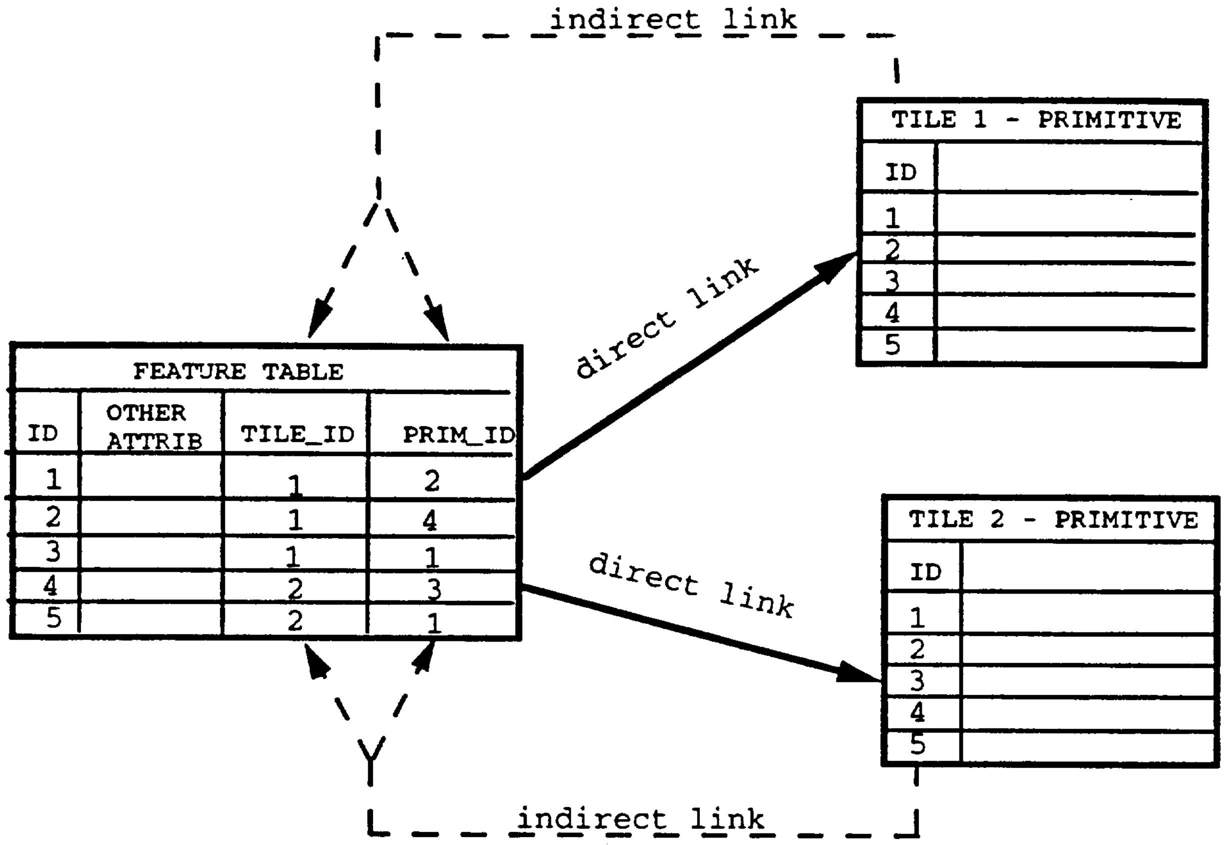


FIGURE 37. Implementation of a 1:1 feature class in a tiled coverage.

• Direct link between feature and primitive provide relatively good performance

• However, indirect links going from the primitive back to the feature provide poor performance.

C.4.3.3 1:1 feature class in a tiled coverage with thematic indexes. Implementation of thematic indexes on the TILE_ID and/or PRIMITIVE_ID columns in the feature table described in C.4.3.2 (FIGURE 38) can improve the performance when going from feature to primitive and is generally recommended. For example, a thematic index on the TILE_ID column in the feature table would provide a list of all records for a given TILE_ID value and thus improve performance on a query of all features within a selected tile. Recall (see C.4.1.3.1), that thematic indexes cannot be implemented on a PRIMITIVE_ID column defined as a data type K (triplet id). Performance of the indirect link between primitive and feature is also enhanced by implementation of thematic indexes on the TILE_ID and PRIMITIVE_ID columns of feature tables. Without indexes on these columns, a sequential search of all records in the feature table would have to be performed to find all matching ids. The indexes improve performance by providing a list of the records with matching ids.

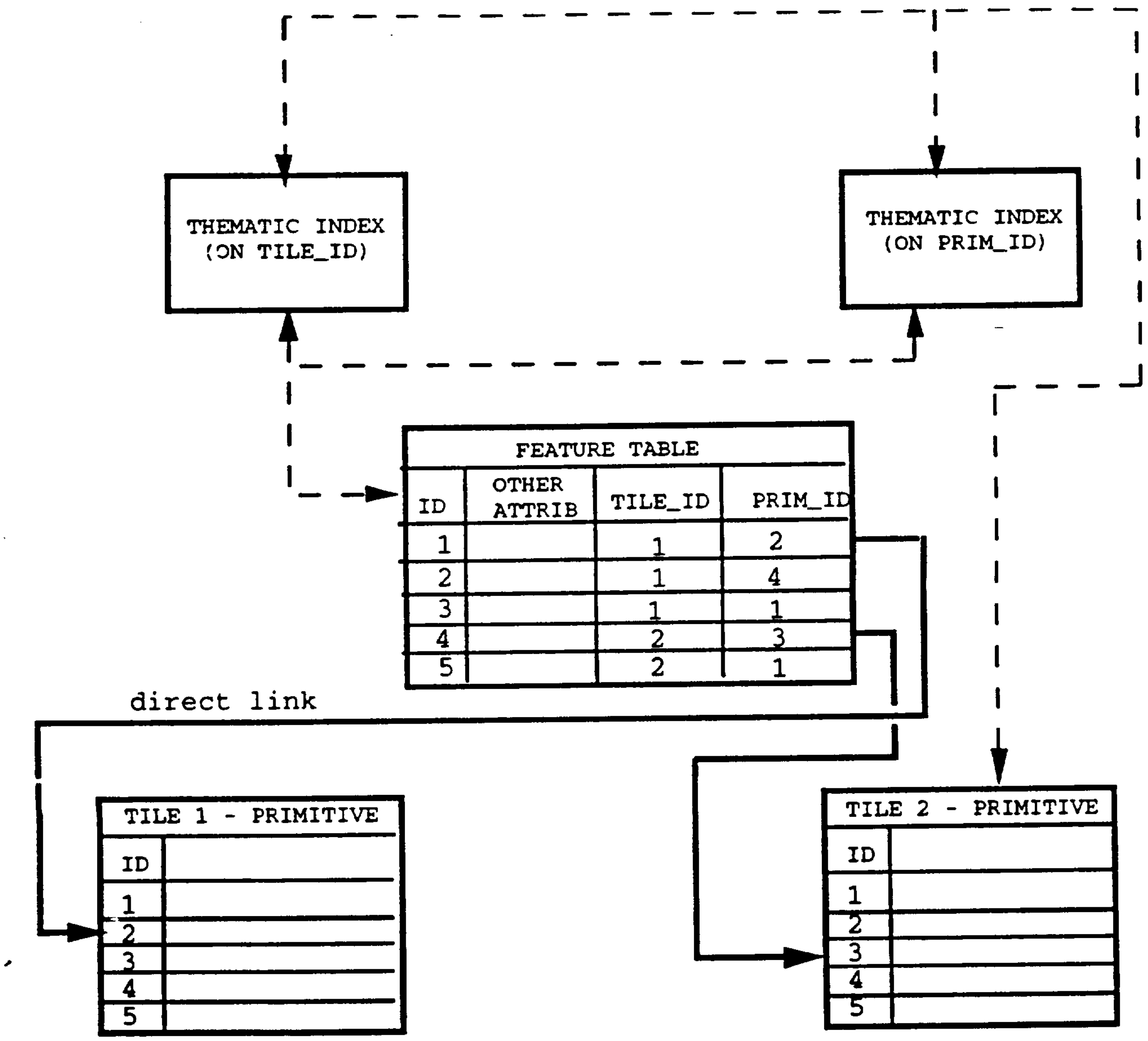
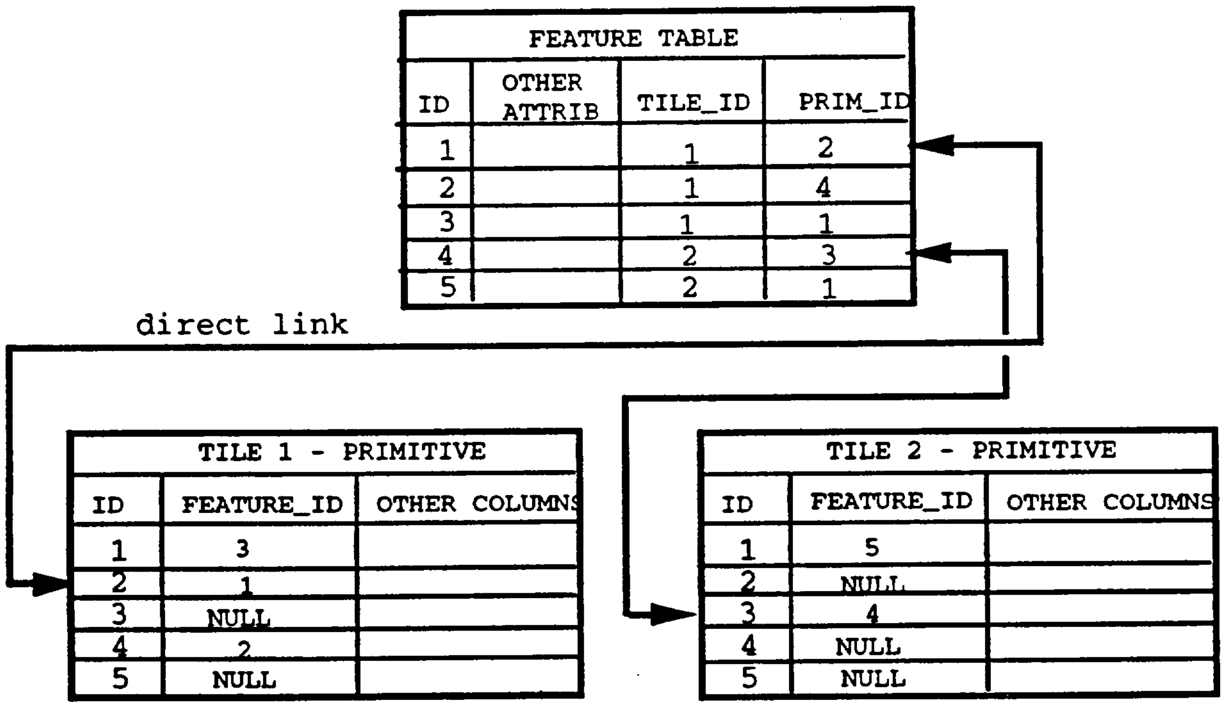


FIGURE 38. Implementation of a 1:1 feature class in a tiled coverage with a thematic index

• Implementation of thematic indexes improves performance of the direct link between feature and primitive.

- Performance of the indirect link between primitive and feature is also improved with the implementation of thematic indexes on the TILE_ID and PRIMITIVE_ID columns in feature tables.

C.4.3.4 <u>1:1 feature classes in a tiled coverage with FEATURE ID pointers in the primitive tables</u>. One way to improve the performance of the implementation described in C.4.3.3 is by adding a FEATURE_ID column to the primitive tables (FIGURE 39) This improves performance by adding a direct link between the primitive and associated feature. However, database designers and software developers should be aware of some of the shortcomings and/or limitations of this feature class design. Implementation of this design in a coverage with multiple feature classes for a given primitive type (i.e. ROADL.LFT, RAILROADL.LFT, TUNNELL.LFT in a Transportation Coverage) results in an unnormalized primitive table (many of the FEATURE_ID columns will contain null values). Secondly, this design does not allow for the existence of coincident features (N:1 relations) in the same feature class (the FEATURE_ID column can only contain one reference to a feature in a given feature class).



FEATURE TABLE

ID	OTHER ATTRIB	TILE_ID	PRIM_ID
1		1	2
2		1	4
3		1	1
4		2	3
5		2	1

TILE 1 - PRIMITIVE

ID	FEATURE_ID	OTHER COLUMNS
1	3	
2	1	
3	NULL	
4	2	
5	NULL	

TILE 2 - PRIMITIVE

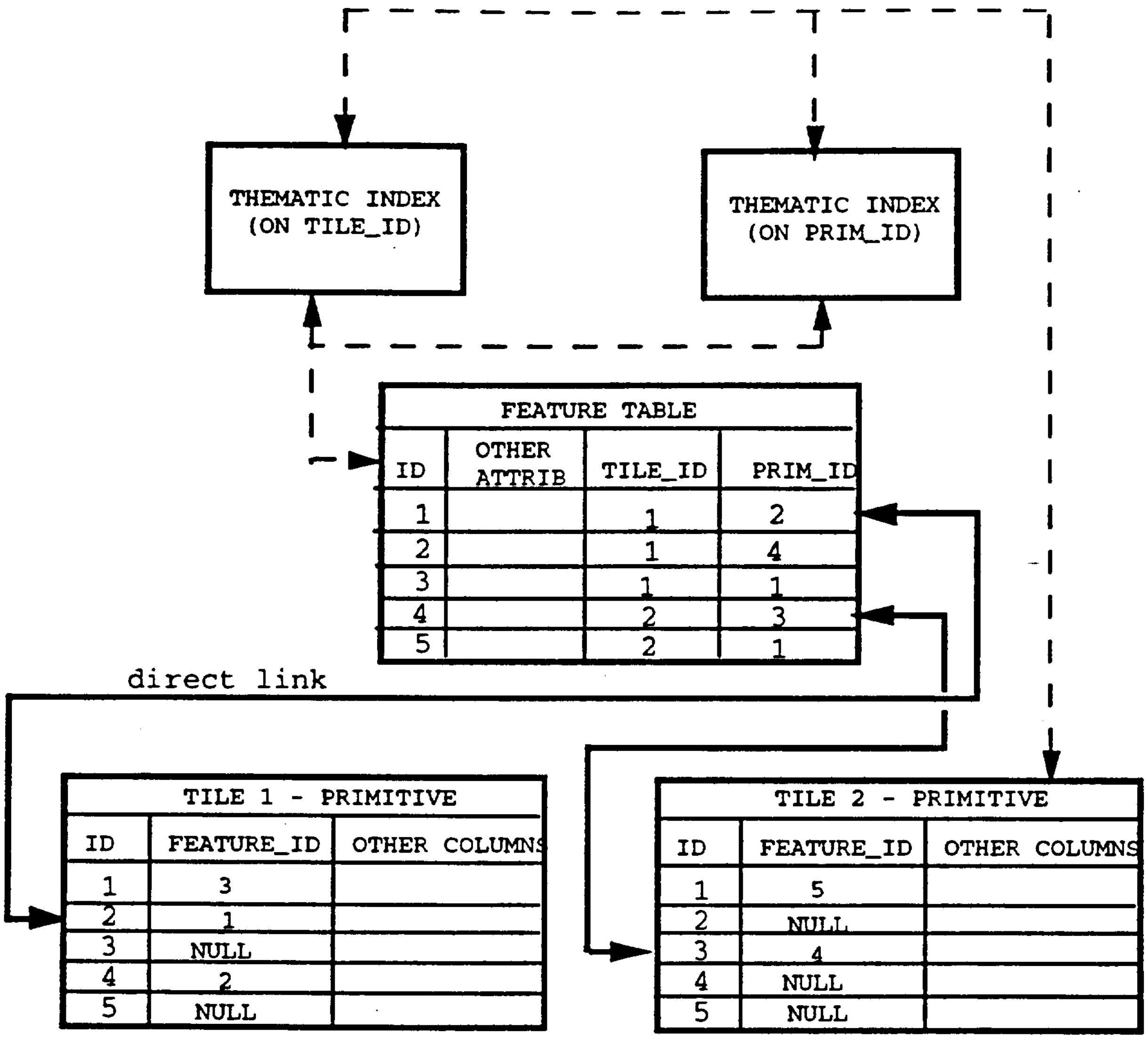
ID	FEATURE_ID	OTHER COLUMNS
1	5	
2	NULL	
3	4	
4	NULL	
5	NULL	

FIGURE 39. <u>Implementation of a 1:1 feature class in a tiled coverage with FEATURE ID columns added to the primitive tables</u>.

- Feature-to-Primitive and Primitive-to-Feature performance is good because of the implementation of direct links

- Unnormalized tables and limitations in coincident features represent disadvantages of this type of design.

C.4.3.5 1:1 feature class in a tiled coverage with FEATURE ID pointers in the primitive tables and thematic indexes. Addition of thematic indexes to the implementation described in C.4.3.4. improves the performance in both the Feature-to-Primitive and Primitive-to-Feature directions (FIGURE 40). See C.4.3.3. for a discussion of how thematic indexes improve performance. Thematic indexes are generally recommended on the TILE_ID and PRIMITIVE_ID columns in feature table(s).



FEATURE TABLE			
ID	OTHER ATTRIB	TILE_ID	PRIM_ID
1		1	2
2		1	4
3		1	1
4		2	3
5		2	1

TILE 1 - PRIMITIVE		
ID	FEATURE_ID	OTHER COLUMNS
1	3	
2	1	
3	NULL	
4	2	
5	NULL	

TILE 2 - PRIMITIVE		
ID	FEATURE_ID	OTHER COLUMNS
1	5	
2	NULL	
3	4	
4	NULL	
5	NULL	

FIGURE 40. Implementation of a 1:1 feature class in a tile coverage with FEATURE ID columns in the primitive tables and thematic indexes.

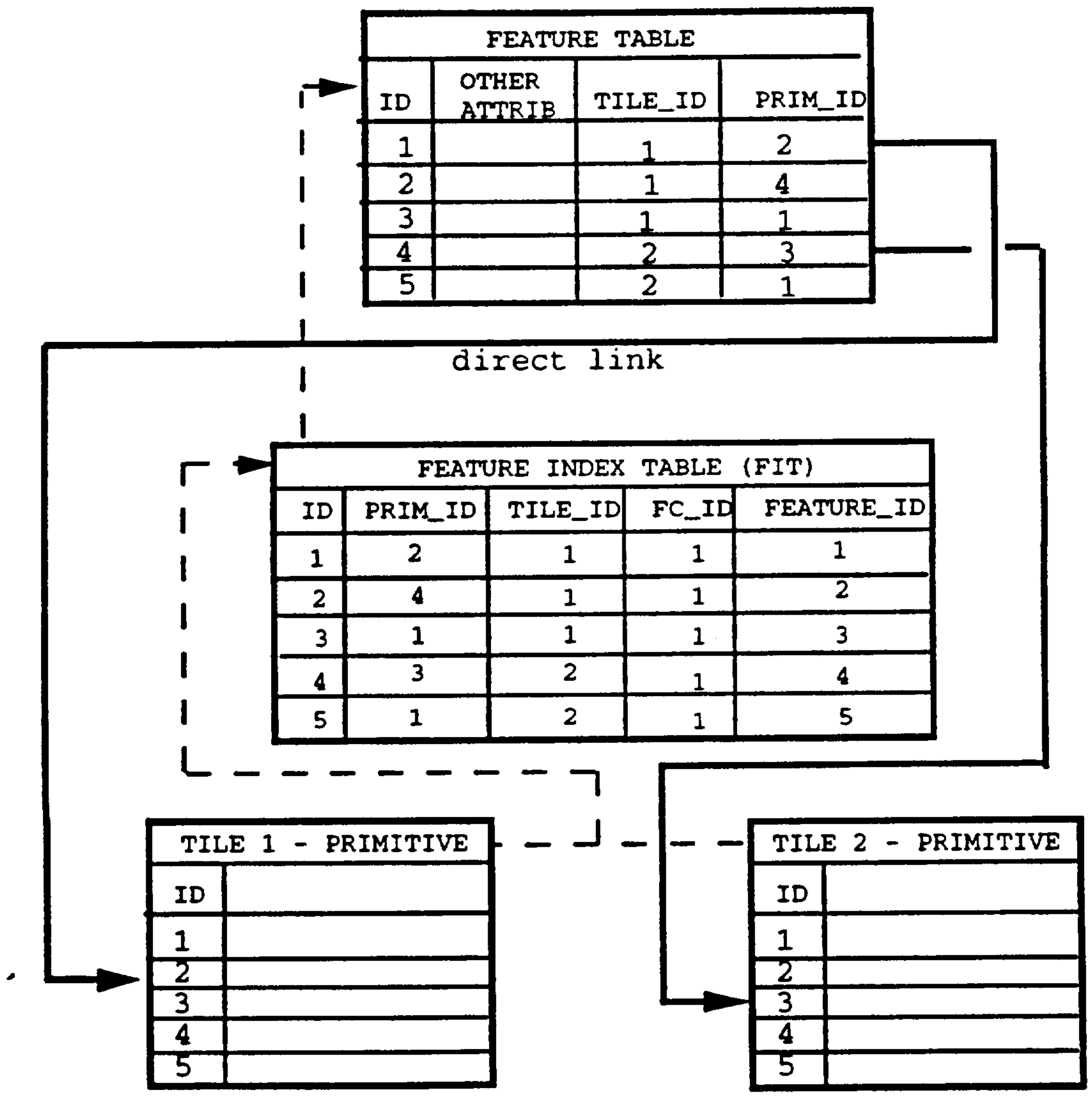
• Feature-to-Primitive and Primitive-to-Feature performance is very good because of the implementation of direct links as well as the addition of thematic indexes

• Unnormalized tables and limitations in coincident features represent disadvantages of this type of design (see C.4.3.4).

C.4.3.6. 1:1 feature class in a tiled coverage with Feature Indexes Implementation of Feature Indexes (FIGURE 41) to improve Primitive-to-Feature performance is an alternative to adding

FEATURE_ID pointers to the primitive tables. This type of implementation is probably best suited for coverages with multiple feature classes of the same primitive type since it eliminates non-normalized primitive tables. It also allows for the existence of coincident features within the same feature class. Implementation of thematic indexes on the following columns in the Feature Index Table (FIT) is recommended: PRIM_ID, TILE_ID, FC_ID, and FEATURE_ID.

FIGURE 40.1 Feature-to-Primitive and Primitive-to-Feature Linkage



FEATURE TABLE

ID	OTHER ATTRIB	TILE_ID	PRIM_ID
1		1	2
2		1	4
3		1	1
4		2	3
5		2	1

FEATURE INDEX TABLE (FIT)

ID	PRIM_ID	TILE_ID	FC_ID	FEATURE_ID
1	2	1	1	1
2	4	1	1	2
3	1	1	1	3
4	3	2	1	4
5	1	2	1	5

TILE 1 - PRIMITIVE

ID	
1	
2	
3	
4	
5	

TILE 2 - PRIMITIVE

ID	
1	
2	
3	
4	
5	

• Feature-to-Primitive and Primitive-to-Feature performance is very good because of the implementation of direct links as well as the addition of a feature index. Performance of this type of design is further enhanced with the implementation of thematic indexes on columns in the Feature Index Table (FIT) and Feature Tables.

• This design eliminates unnormalized primitive tables and allows the creation of coincident features in the same feature class.

C.4.4 <u>One-to-many relationships</u>. The next relation type is one in which a single feature is composed of many primitives. This type of feature is often referred to as a compound feature and requires implementation of a join table to define the one-to-many relationship between a feature and its associated primitives.

C.4.4.1 <u>1:N feature class in an untiled coverage using join tables</u>. As stated in C.4.4, a join table is required to define the one-to-many relationship between a feature and its associated primitives. The PRIMITIVE_ID column in the join table acts as a foreign key and provides a direct link to the primitive(s) associated with a selected feature. The reverse relationship (primitive-to-feature) is an indirect link and provides poor performance. For example, the PRIMITIVE_ID column of all records in all feature join tables would have to be examined to find the feature associated with a selected primitive.

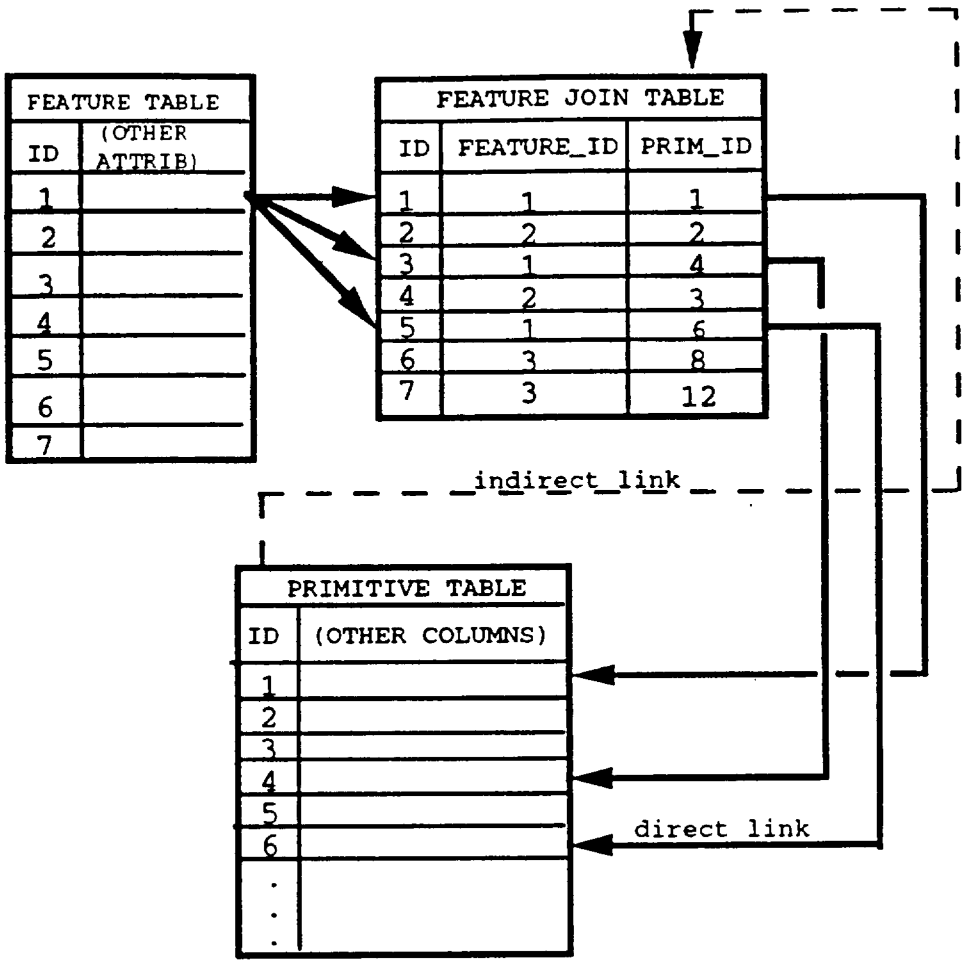


FIGURE 41. Implementation of 1:N feature class in an untiled coverage with a join table.

• Although a direct link between feature and primitive(s) is provided by the PRIMITIVE_ID column in the join table, a sequential search of the FEATURE_ID column must still be performed to find all primitives associated with a selected feature. As a result of the sequential search, performance going from the feature to primitive is relatively slow.

• The indirect link going from the primitive back to the feature provides poor performance.

C.4.4.2 1:N feature class in an untiled coverage with a thematic index Addition of a thematic index to the implementation described C.4.4.1 can improve the performance when going from feature-to-primitive. For example, a thematic index on the FEATURE_ID column in the join table will improve performance on a query which needs to find all primitives associated with a selected feature. Note that primitive to feature performance is still poor because of the indirect link.

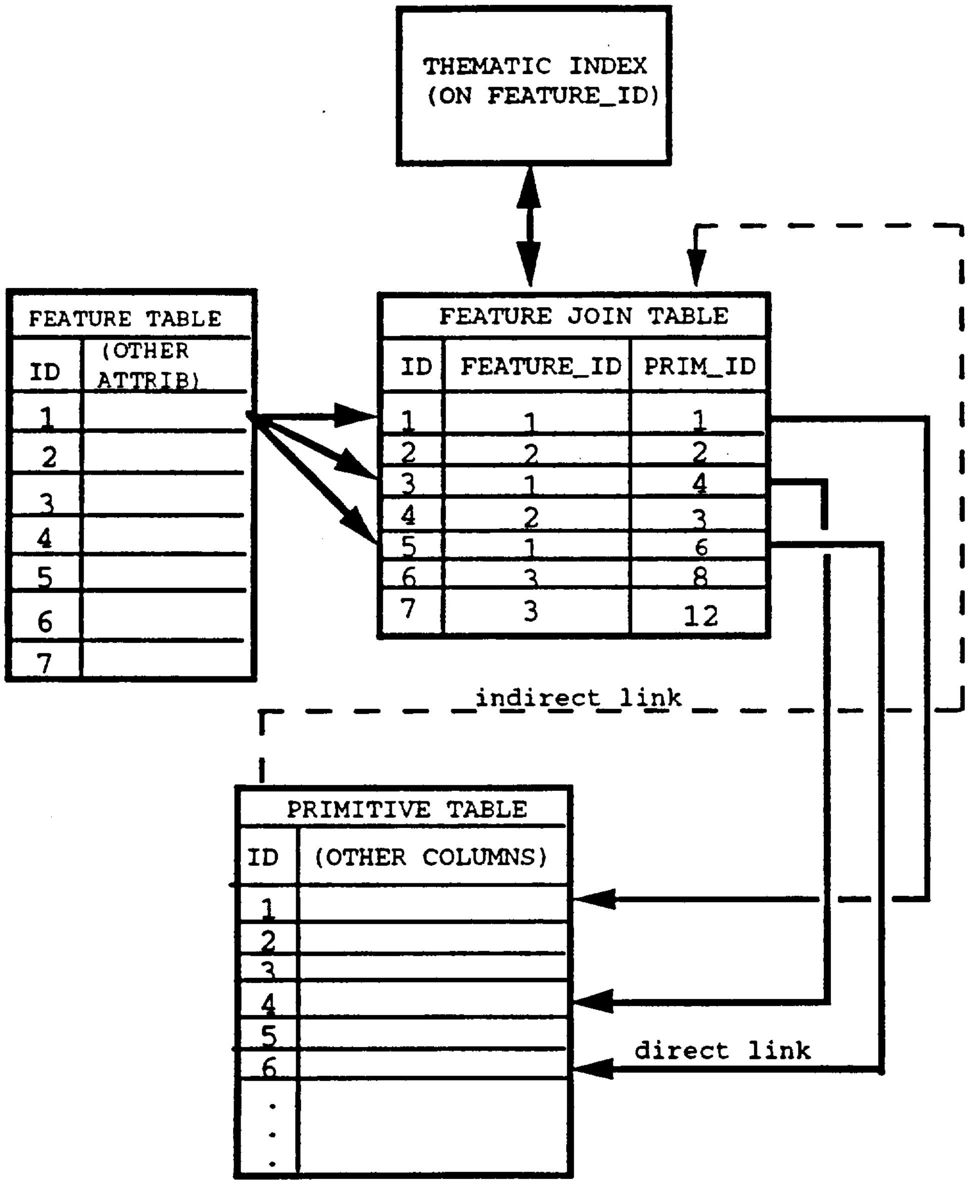
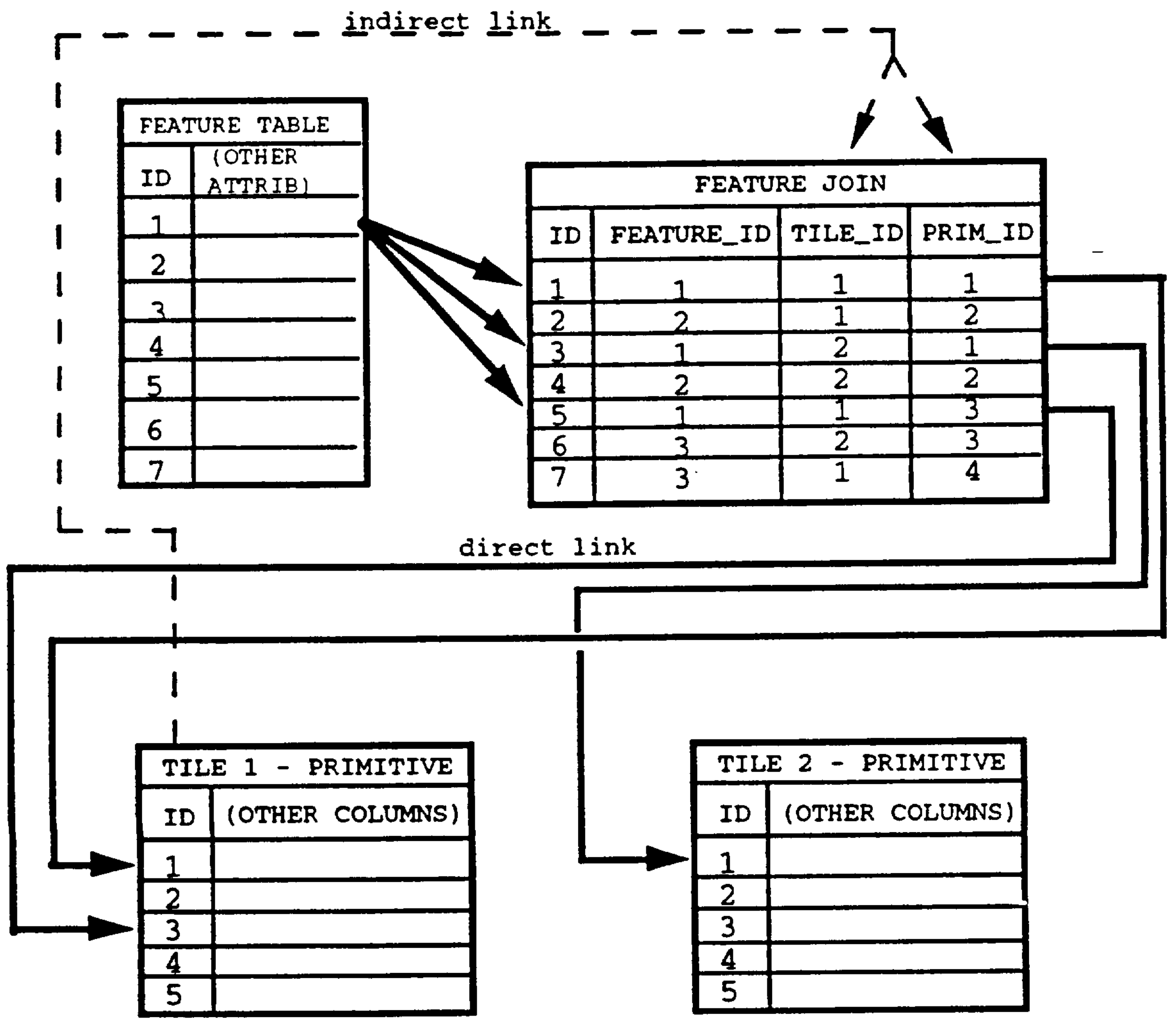


FIGURE 42. Implementation of a 1:N feature class in an untiled coverage using join tables and thematic indexes.

• Addition of a thematic index to the FEATURE_ID column in the join table improves the performance of this feature class design when going from the feature to its primitive(s).

• However, the indirect link going from the primitive back to the feature provides poor performance.

C.4.4.3 <u>1:N feature class in a tiled coverage</u>. The addition of tiles in a 1:N relationship makes for a complex implementation. The implementation shown in FIGURE 43 is conceptually clean but will perform poorly with software because of the lack of thematic indexes.



FEATURE TABLE

ID	(OTHER ATTRIB)
1	
2	
3	
4	
5	
6	
7	

FEATURE JOIN

ID	FEATURE_ID	TILE_ID	PRIM_ID
1	1	1	1
2	2	1	2
3	1	2	1
4	2	2	2
5	1	1	3
6	3	2	3
7	3	1	4

TILE 1 - PRIMITIVE

ID	(OTHER COLUMNS)
1	
2	
3	
4	
5	

TILE 2 - PRIMITIVE

ID	(OTHER COLUMNS)
1	
2	
3	
4	
5	

FIGURE 43. <u>Implementation of a 1:N feature class in a tiled coverage</u>.

• Although a direct link between feature and primitive(s) is provided by the TILE_ID and PRIMITIVE_ID columns in the join table, a sequential search of the FEATURE_ID column must still be performed to find all primitives associated with a selected feature. As a result of the sequential search, performance going from the feature to primitive is relatively slow.

• The indirect link going from the primitive back to the feature provides poor performance.

C.4.4.4 Tiled 1:N coverages with FEATURE ID pointers in the primitive tables. The addition of a FEATURE_ID column to the primitive tables (FIGURE 44) provides good performance from the primitive to the feature. However, keep in mind the disadvantages and/or shortcomings of this method as identified in C.4.3.4. Performance from the feature to the join table to the primitive is relatively poor without the implementation of thematic indexes.

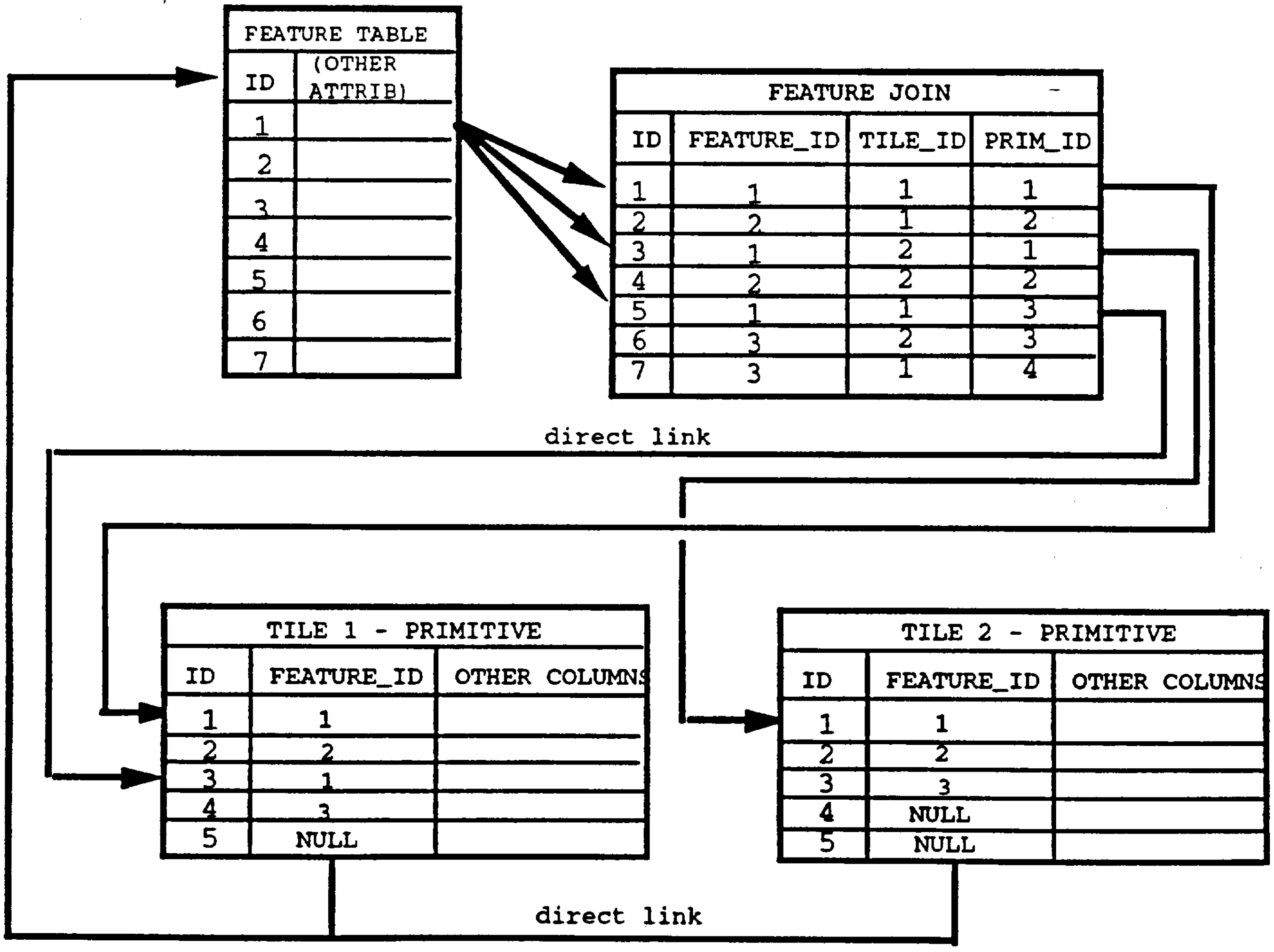


FIGURE 44. Implementation of a 1:N feature class in a tiled coverage that includes FEATURE ID columns in primitive tables.

- Primitive-to-Feature performance is good because of the implementation of direct links. However, be aware of the shortcomings and disadvantages of the implementation of FEATURE_ID pointers in primitive tables (see C.4.3.4)

- Although a direct link between feature and primitive(s) is provided by the TILE_ID and PRIMITIVE_ID columns in the join table, a sequential search of the FEATURE_ID column must still be performed to find all primitives associated with a selected feature. As a result of the sequential search, performance going from the feature to primitive is relatively slow.

C.4.4.5 1:N feature class in a tiled coverage with FEATURE ID pointers in the primitive tables and thematic indexes. The addition of thematic indexes to columns in the join table (FIGURE 45) improves performance going from the feature to primitive(s). Thematic indexes are generally recommended on the FEATURE_ID and TILE_ID columns in join tables. Implementation of FEATURE_ID pointers in primitive tables provides good performance going from the primitive to feature. However, keep in mind the disadvantages and/or shortcomings of this method as identified in C.4.3.4.

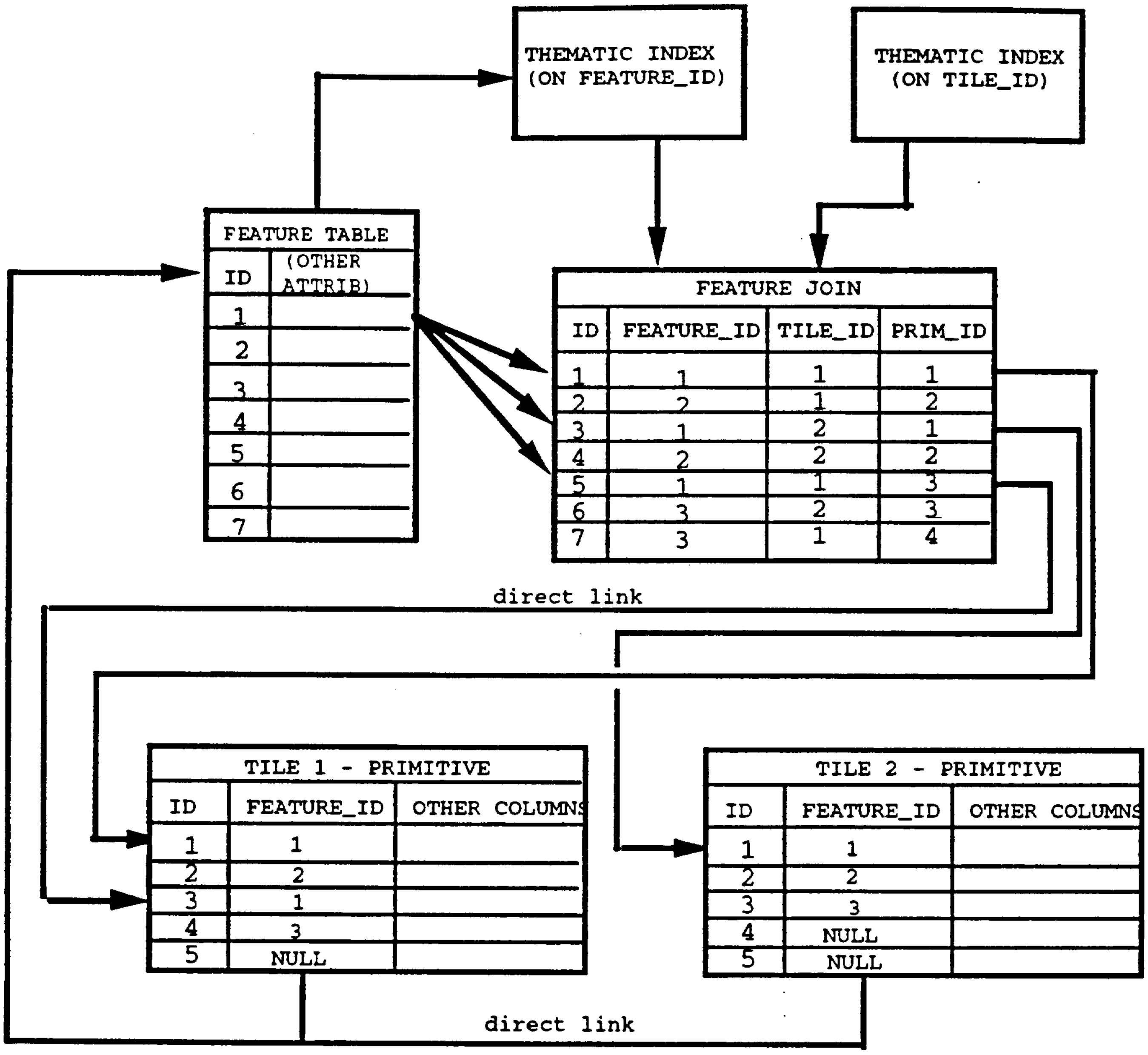
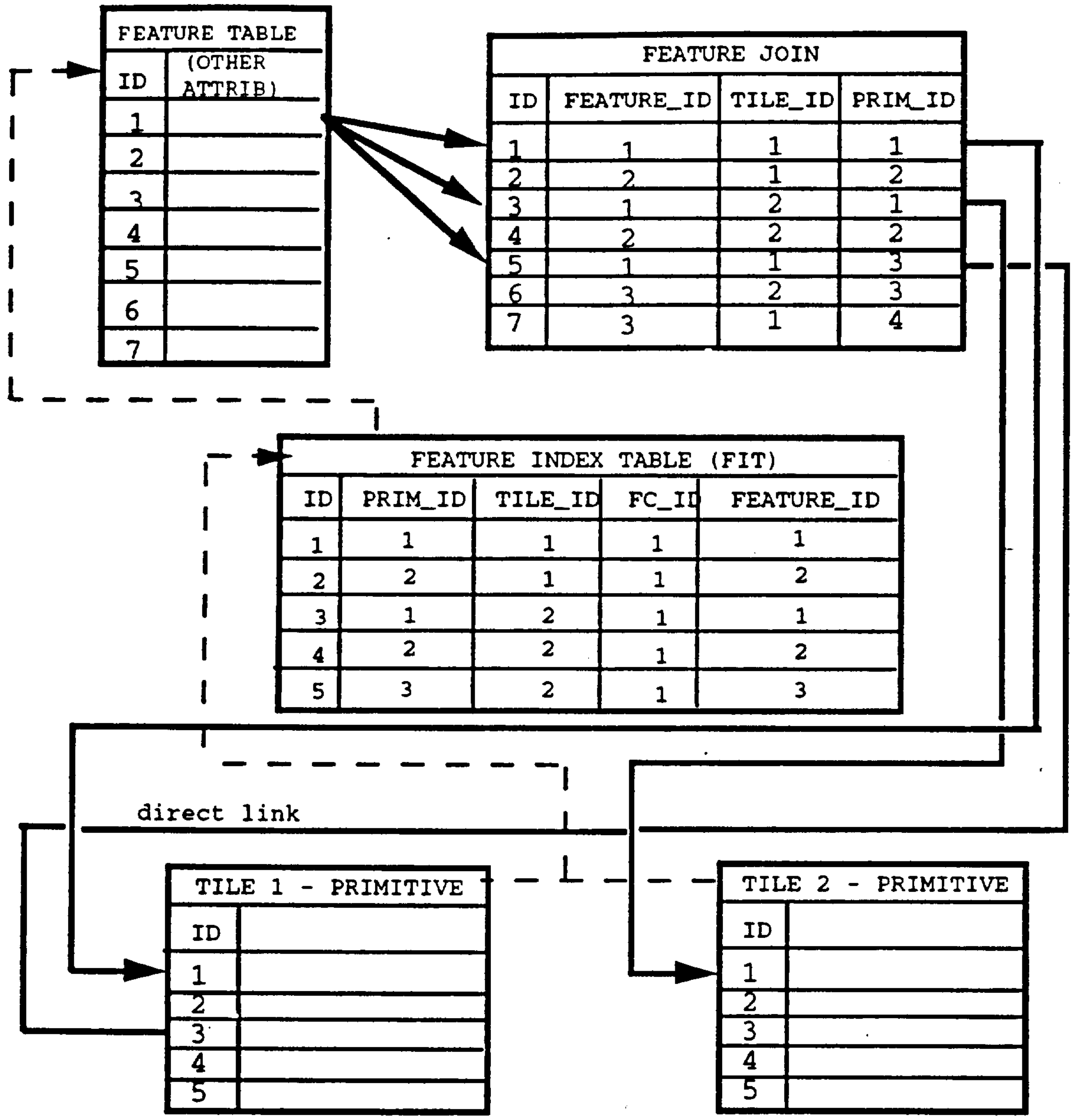


FIGURE 45. Implementation of a 1:N feature class in a tiled coverage that includes FEATURE ID columns and thematic indexes.

• Feature-to-Primitive and Primitive-to-Feature performance is good because of the implementation of direct links as well as the addition of thematic indexes

- Be aware of the shortcomings and disadvantages of the implementation of FEATURE_ID pointers in primitive tables (see C.4.3.4)

C.4.4.6 1:N feature class in a tiled coverage with feature indexes and thematic indexes. As stated in C.4.3.6, implementation of feature indexes to improve Primitive-to-Feature performance is an alternative to adding FEATURE_ID pointers to primitive tables. It is probably best suited for coverages with multiple feature classes of the same primitive type since it eliminates non-normalized primitive tables. Implementation of thematic indexes on the FEATURE_ID and TILE_ID column in the join tables as well as the following columns in the Feature Index Tables (FITs) is generally recommended: PRIM_ID, TILE_ID, FC_ID, and FEATURE_ID.



FEATURE TABLE	
ID	(OTHER ATTRIB)
1	
2	
3	
4	
5	
6	
7	

FEATURE JOIN			
ID	FEATURE_ID	TILE_ID	PRIM_ID
1	1	1	1
2	2	1	2
3	1	2	1
4	2	2	2
5	1	1	3
6	3	2	3
7	3	1	4

FEATURE INDEX TABLE (FIT)				
ID	PRIM_ID	TILE_ID	FC_ID	FEATURE_ID
1	1	1	1	1
2	2	1	1	2
3	1	2	1	1
4	2	2	1	2
5	3	2	1	3

TILE 1 - PRIMITIVE	
ID	
1	
2	
3	
4	
5	

TILE 2 - PRIMITIVE	
ID	
1	
2	
3	
4	
5	

- Feature-to-Primitive performance is good because of the direct link and thematic indexes.

- Primitive-to-Feature performance is good because of the implementation of feature indexes.

C.4.5 <u>Many-to-one relationships</u>. When multiple features are associated with the same primitive, the relationship is often referred to as coincident features or N:1. For the sake of brevity, no examples of N:1 relations are provided. Instead, refer to section C.4.3 (1:1 relations) for the different types of performance enhancers available, keeping in mind the shortcomings/disadvantages identified in C.4.3.4.

C.4.6 Many-to-many relationship. In many-to-many (N:M) relationships, many features can relate to many primitives and the reverse. Many features can relate to one primitive, and many primitives can relate back to one feature. This relation is established using a join table. Many-to-many relations are typical and often times unavoidable in an integrated coverage database design. Again, for the sake of brevity, no examples are provided for N:M relations. Instead, refer to sections C.4.3 and C.4.4 for the type of performance enhancers available, making note of the advantages and disadvantages of each.

C.4.7 Complex feature relationships. A complex feature may be constructed from simple features only, or from other complex features. This forms a hierarchical feature relationship. The need for complex features arises when a group of features requires different attributes than that of other features.

C.4.7.1 One complex feature composed of simple features from multiple feature classes. As FIGURE 46 shows, a complex feature can be composed of simple features from multiple feature classes (in this case, an area feature class and line feature class). The implementation shown in FIGURE 46 is useful when not all simple features are part of a complex feature or when the complex feature is created after the simple features. This type of complex feature design has its limitations, however. Because of the implementation of FEATURE_ID foreign keys (i.e. *.AFT_ID and *.LFT_ID) in the complex feature table, a complex feature cannot be composed of more than one simple feature from a given feature class. For example, this type of design does not support the creation of an Interstate Hwy. complex feature composed of a number of road line simple features. Implementation of a complex feature join table which defines the complex feature to simple feature relations is generally considered a better design (see C.4.7.2.)

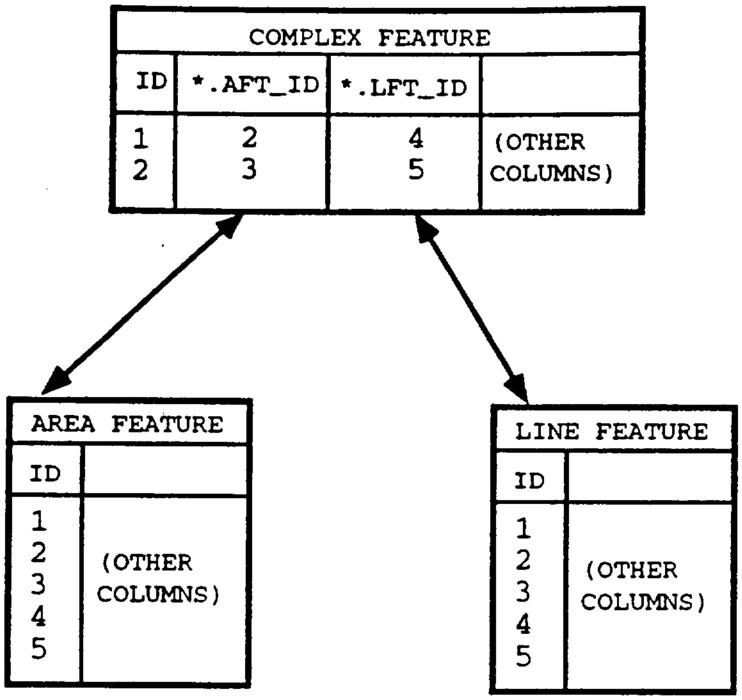
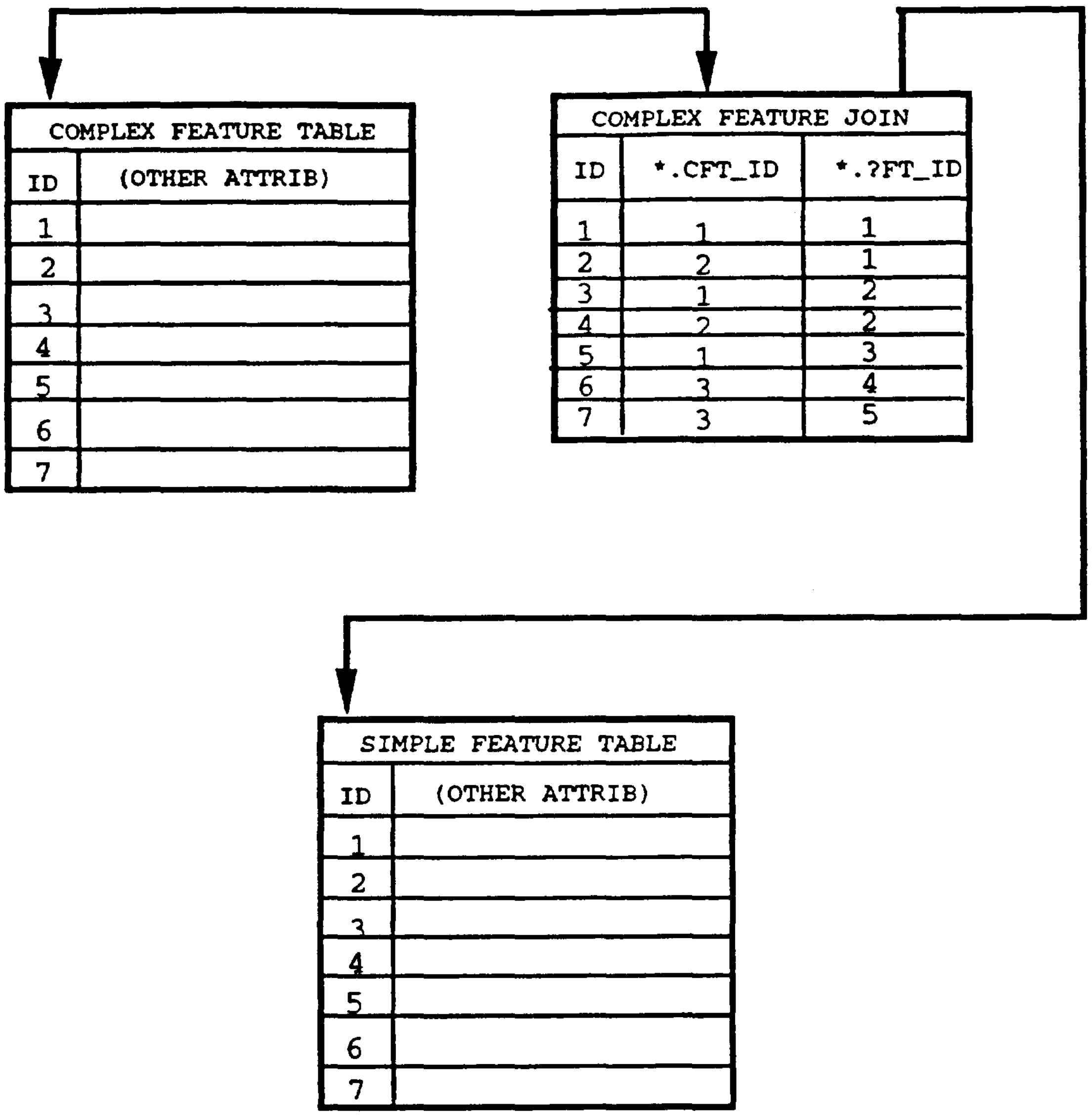


FIGURE 46. Implementation of a complex feature composed of simple features in separate tables.

C.4.7.2 Many complex, many simple features. Many complex features may be made up of many simple features in one feature table. The implementation shown in FIGURE 47 requires complex features and simple features to be created at the same time. It also requires the implementation of a complex feature join table to describe the complex feature to simple feature relations. Performance can be improved by adding thematic indexes on both the columns in the join table.



COMPLEX FEATURE TABLE	
ID	(OTHER ATTRIB)
1	
2	
3	
4	
5	
6	
7	

COMPLEX FEATURE JOIN		
ID	*.CFT_ID	*.?FT_ID
1	1	1
2	2	1
3	1	2
4	2	2
5	1	3
6	3	4
7	3	5

SIMPLE FEATURE TABLE	
ID	(OTHER ATTRIB)
1	
2	
3	
4	
5	
6	
7	

FIGURE 47. Implementation of a complex feature relationship in which many complexfeatures are made up of many simple features in one feature table.

C.4.7.3 An example of a coverage with simple and complex features. TABLE 68 is the feature class schema table from the Terminal Procedure Coverage of DFLIP Prototype No. 2. The terminal procedure complex feature (termpc) consists of action points (pactc) which are related to connected nodes, Instrument Landing System (ILS) line features (ilsl) and other route line features (seg2l). Records 3 and 4 show the relationships for the action points (pactc) and their connected nodes (cnd), records 7 to 10 show how the ILS lines (ilsl) are joined to multiple edges, and records 15 to 18 show how route lines (seg2l) are joined to multiple edges. The complex feature (termpc) realtionships begin with record 19 and continue through record 40. FIGURE 48 is helpful in tracing each record to the first and second table columns of the record.

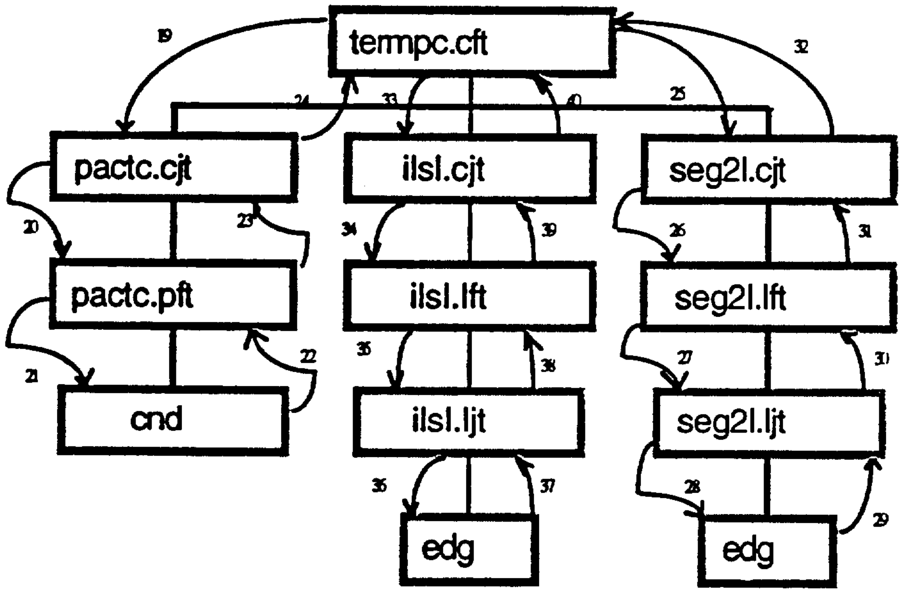


FIGURE 48. FCS record numbers linking tables for complex feature.

TABLE 68. Content and Format for Terminal Procedures Coverage Feature Class Schema Table.

```
{Header length}L;
Terminal Procedures Feature Class Schema Table;-;
id=I,1,P,Row Identifier,-,-,-,:
feature_class=T,7,N,Name of Feature Class,-,-,-,:
table1=T,12,N,First Table in a Relationship,-,-,-,:
table1_key=T,16,N,Column Name in First Table,-,-,-,:
table2=T,12,N,Second Table in a Relationship,-,-,-,:
table2_key=T,16,N,Column Name in Second Table,-,-,-,:;
```

1	nav2c	nav2c.pft	cnd_id	cnd	id
2	nav2c	cnd	id	nav2c.pft	cnd_id
3	pactc	pactc.pft	cnd_id	cnd	id
4	pactc	cnd	id	pactc.pft	cnd_id
5	hold2c	hold2c.pft	cnd_id	cnd	id
6	hold2c	cnd	id	hold2c.pft	cnd_id
7	ilsl	ilsl.lft	id	ilsl.ljt	ilsl.lft_id
8	ilsl	ilsl.ljt	edg_id	edg	id
9	ilsl	edg	id	ilsl.ljt	edg_id
10	ilsl	ilsl.ljt	ilsl.lft_id	ilsl.lft	id
11	rab2l	rab2l.lft	id	rab2l.ljt	rab2l.lft_id
12	rab2l	rab2l.ljt	edg_id	edg	id
13	rab2l	edg	id	rab2l.ljt	edg_id
14	rab2l	rab2l.ljt	rab2l.lft_id	rab2l.lft	id
15	seg2l	seg2l.lft	id	seg2l.ljt	seg2l.lft_id
16	seg2l	seg2l.ljt	edg_id	edg	id
17	seg2l	edg	id	seg2l.ljt	edg_id
18	seg2l	seg2l.ljt	seg2l.lft_id	seg2l.lft	id
19	termpc	termpc.cft	id	pactc.cjt	termpc.cft_id
20	termpc	pactc.cjt	pactc.pft_id	pactc.pft	id
21	termpc	pactc.pft	cnd_id	cnd	id
22	termpc	cnd	id	pactc.pft	cnd_id
23	termpc	pactc.pft	id	pactc.cjt	pactc.pft_id
24	termpc	pactc.cjt	termpc.cft_id	termpc.cft	id
25	termpc	termpc.cft	id	seg2l.cjt	termpc.cft_id
26	termpc	seg2l.cjt	seg2l.lft_id	seg2l.lft	id
27	termpc	seg2l.lft	id	seg2l.ljt	seg2l.lft_id
28	termpc	seg2l.ljt	edg_id	edg	id
29	termpc	edg	id	seg2l.ljt	edg_id
30	termpc	seg2l.ljt	seg2l.lft_id	seg2l.lft	id
31	termpc	seg2l.lft	id	seg2l.cjt	seg2l.lft_id
32	termpc	seg2l.cjt	termpc.cft_id	termpc.cft	id
33	termpc	termpc.cft	id	ilsl.cjt	termpc.cft_id
34	termpc	ilsl.cjt	ilsl.lft_id	ilsl.lft	id
35	termpc	ilsl.lft	id	ilsl.ljt	ilsl.lft_id
36	termpc	ilsl.ljt	edg_id	edg	id
37	termpc	edg	id	ilsl.ljt	edg_id
38	termpc	ilsl.ljt	ilsl.lft_id	ilsl.lft	id
39	termpc	ilsl.lft	id	ilsl.cjt	ilsl.lft_id
40	termpc	ilsl.cjt	termpc.cft_id	termpc.cft	id

APPENDIX D

TILING

D.1. GENERAL

D.1.1 Scope. This appendix provides information and discussion concerning tiling for a VPF database. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

D.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

D.3. DEFINITIONS

D.3.1 Definitions used in this appendix. For purposes of this appendix, the definitions of section 3 of the main document shall apply.

D.40. GENERAL INFORMATION

D.40.1 Rationale. Global scale databases inevitably consist of large amounts of data. In processing geographic data, entire files often need to be managed in memory, imposing a definite limit on database size. Tiling is the method used to break up geographic data into spatial units small enough to fit within the limitations of the desired hardware platform and media. VPF libraries are partitioned into a tile structure defined in a particular product specification and TILEREF coverage. The naming convention is based on the geographic reference system GEOREF. The actual tile size is a product-specific question dependent upon the minimum hardware configuration and distribution media. The following paragraphs describe how VPF implements tiling to subdivide a database.

D.40.2 Cross-tile topological primitives. One shortcoming of past tiling implementations occurs when primitives are split up into different files, which also removes the topological connectivity of the feature. If a lake feature's primitive is split into two separate tiles, the topological connection between the primitives of the lake is lost. They will still appear together when both tiles are drawn on the screen, but any analysis that tries to follow the original connectivity of the lakeshore will have to do a lot of extra processing and searching. It would thus be advantageous for the primitives of the tiled lake shoreline to refer to each other. There is a need for primitives that cross tile boundaries to refer both to the tile boundary itself (in order to maintain tile topology) and to its cross-tile continuation in order to simplify retrieval of the original feature. VPF meets this need by referring to edges using a triplet id that contains an internal reference to a boundary or edge within the current tile; when appropriate, the triplet id also contains an external reference to an edge in a neighboring tile.

D.40.3 Feature classes. Tiling introduces a constraint on feature classes as well as primitives. Tiling is a low-level implementation issue related to hardware and storage limitations, and should have no effect upon conceptual structures like feature classes. Unfortunately, when primitives are broken up into separate tables for tiling, their corresponding attributes are broken up as well. VPF resolves this by maintaining the attribute tables and feature class tables unbroken and structuring them so that they can be processed sequentially rather than all at once, thus obviating the need to break them up to meet hardware limits. These tables are stored within a coverage, and the actual file subdivisions representing the tiles appear as directories underneath the coverage. The problem concerns connecting a primitive, now stored in one of many smaller files, to its attributes, now stored in a single large file. This is done in accord with standard relational design rules by adding a column for each feature class onto the primitive file, containing the row id from the master table that has the attributes for that primitive. If five feature classes are derived from the primitive topological layer, then five extra columns will be added to that primitive table. Adding another column for each feature class could make for a very large table since there are 468 FACC feature classes. Not all feature classes apply to all three primitive dimensions or to all products, or to all coverages, but for vertically or thematically integrated data the number could still easily approach 100. The reason for a pointer back to the feature is merely for performance issues. Please refer to feature class construction issues in APPENDIX C.

VPF handles tiling and data partition problems at the primitive level by means of the triplet id to maintain cross-tile topology and the extra feature class columns on the primitive to maintain links to the more logically consistent single attribute table. There is also the need to maintain the tiles themselves and to store reference data about the tiles, their size, scheme, and so forth. This is accomplished with the tile reference coverage.

D.40.4 Tile reference coverage. The tile reference coverage is a polygon coverage representing only the tiles. No other data besides tile boundaries and tile labels is used. This is stored as a separate coverage at the library level and acts as a graphic index to the tiling scheme, showing all of the tiles and only those tiles in the library, their names, and their relation to each other. The area feature table of this coverage plays a very important role. Since it can store attributes about each tile, it can be used to hold data density figures, tile data volume, summary contents for each feature class, and other metadata to assist in managing the database at a coarse tile-by-tile level. The tile size, actual tile layout, and handling of text that crosses tile borders are not addressed in this standard since they are all product-specific questions.

APPENDIX E

DATA QUALITY

E.1. GENERAL

E.1.1 Scope. This appendix provides information, discussion, and examples concerning data quality issues in a VPF database. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

E.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

E.3. DEFINITIONS

E.3.1 Definitions used in this appendix. For purposes of this appendix, the definitions of section 3 of the main document shall apply.

E.4. GENERAL INFORMATION

E.4.1 VPF data quality. This appendix describes basic strategies for storing data quality (DQ) information within VPF databases. The following subsections discuss general data quality concepts, implementation of the data quality table, and data quality coverages within VPF.

E.4.2 General concepts. The multilevel structure (i.e., primitive, feature class, coverage, library, and database) of a VPF database affects the strategies used to store DQ information. DQ information is often available at varying levels of specificity, from individual feature attributes to expressions of quality germane to an entire database. Therefore, it is necessary for the VPF data producer to determine the appropriate level in the hierarchy for the various types of DQ information present. When developing an implementation strategy, the data producer should also review the available DQ information within the context of coverage (thematic) associations as well as spatial extent, as these will influence the use of standard data quality tables and/or the development of separate data quality coverages.

When DQ information is stored at multiple levels in a VPF database, lower level information always takes precedence over that at higher levels. Data producers should account for this when compiling DQ information to be stored in VPF. For example, a data producer may make a general statement at the library level that a coverage has been derived from a range of sources, and specify at the primitive level what a given feature's exact origins are. Alternatively, some variations may be organized by simple spatial divisions, such as source map boundaries. In such a case, feature level source information would be highly repetitious, and a data quality coverage might be most effective.

E.4.3 Data quality tables. The data quality table (TABLE 47) is a device for storing standard and nonstandard types of DQ information. Standard information is explicitly defined within the DQ table. Nonstandard information is that which is not accounted for in the DQ table and must be stored outside the standard fields. The DQ table can be implemented on the database, library, or coverage level, depending upon the uniformity of the affected data with respect to known DQ characteristics. Key characteristics are source, positional accuracy, attribute accuracy, logical consistency, completeness, resolution, and lineage. All of these characteristics can be documented within standard DQ table fields, with the exception of lineage.

E.4.3.1 Lineage. Lineage information must be stored within the DQ table narrative file, which should be given the name "LINEAGE.DOC." The lineage file is an important component of the DQ documentation system, since the data producer must inevitably make key decisions affecting the data's fitness for use that cannot be described in standard fields. Lineage information may also change significantly from coverage to coverage, even when all of the data is derived from a single source. At a minimum, the lineage file should contain information on processing tolerances, interpretation rules applied to source materials, and basic production and quality assurance procedures. All lineage information available through the source should also be incorporated here.

E.4.3.2 Placement of DQ table. When implementing DQ tables, the user must determine at what level within the VPF hierarchy the DQ table(s) should be established. This will vary depending upon the specific nature of the DQ information. For example, entire libraries originating from a single source may be best served with a single table at that level, augmented by a series of data producer-defined tables implemented at the coverage level to document more specific information. The DQ table can also be implemented on multiple levels simultaneously, with general (broad) information provided at the higher levels, and progressively more detailed information specified at the lower levels. At a minimum, some form of DQ information should be present at the database or library levels to provide an overview of characteristics and to describe the techniques used to store DQ information within the database as a whole. With respect to VPF tables in general, the data producer is not restricted to the standard DQ table. The table can be augmented as needed with producer-defined related files within the hierarchy.

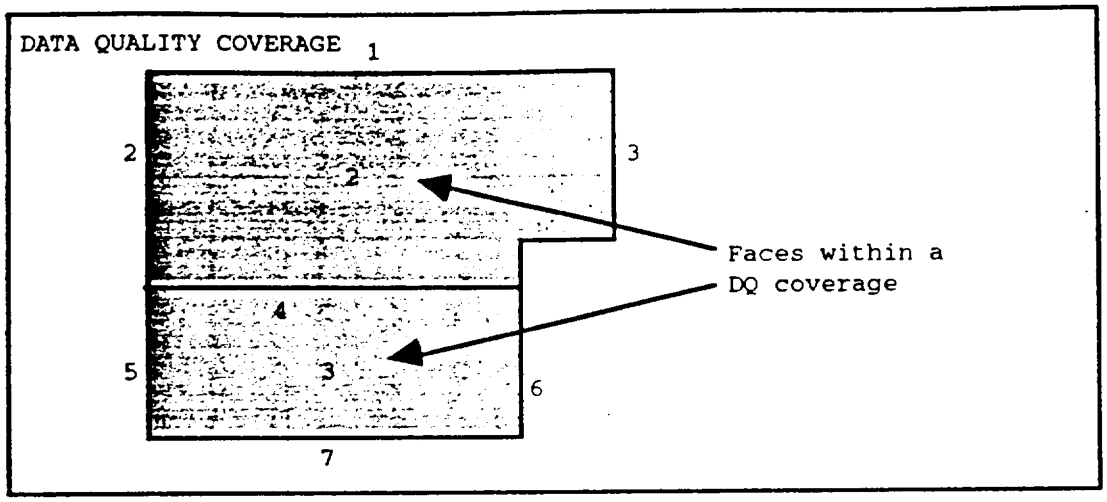
E.4.4 Data quality coverages. Data quality coverages delineate spatial variations in DQ information across a database or a library by assigning unique quality characteristics to areas. Within this context, the database and library levels must be viewed as having distinct spatial extents to which attributes can be assigned, while data quality coverages are created to document spatial variations on a more detailed level. DQ coverages are very useful as visual reference data that allow data producers to capture anomalous data behavior within the context of known spatial variations. They are, by definition, level 3 topology coverages. They can be physically stored at any level (above the feature level) within the VPF hierarchy, independent of the level of information being represented. The manner in which the DQ information is structured within the coverage will be dependent upon the relationships between spatial variations, data sources, and lineage information. Edges, as well as areas, can also be attributized within DQ coverages, for example, to document the interaction of disparate data sets that are not well reconciled. A more specific example is where contour lines from different data sources meet along a common edge and fail to match positionally. In this case, the data producer could document this discrepancy as an attribute of the seam (edge) between the adjacent sources.

Unlike tabular DQ information, the data producer should refrain from creating "nested" DQ coverages at multiple levels, as these can greatly complicate the interpretation of the information. Rather, the user is encouraged to adopt a more integrated approach, where general information is carried on lower level polygons in addition to level-specific information.

E.4.4.1 Coverage components. VPF provides a variety of mechanisms for storing DQ information within coverages, including attribute tables, standard data quality tables, and optional user-defined relational tables. The mechanisms employed vary depending primarily upon the types of information being stored, rather than the VPF level at which it will reside. DQ coverage attribute tables offer the highest degree of flexibility in storing DQ information, since users can design their own table formats and specifically code those components that vary spatially across the data set. Standard DQ tables (TABLE 46) are particularly useful for data sets where characteristics change radically (with respect to source) from one area to another. In this application, data quality tables are stored as multiple records, with each complete DQ table record corresponding to a face. Within this context, it may be particularly appropriate for data producers to implement subsets or supersets of the standard DQ table. Additional relational tables are useful in normalizing complex data, a technique that is particularly helpful when addressing thematically based variations in DQ information (section E.4.4.2.2 of this appendix). Finally, text information is useful for describing phenomena without well-defined spatial extents or for annotating special conditions that do not occur with sufficient frequency to justify creating attribute fields to describe them.

E.4.4.2 Coverage examples. The following sections provide examples of how DQ coverages may be designed under a variety of conditions. The user is encouraged to adopt these approaches when appropriate and to modify them when necessary. Whenever possible, the producer should use coverage level descriptive tables to document the strategy employed in designing and developing DQ coverages for a database.

E.4.4.2.1 Shared regions and common attributes. The most simple DQ coverage is one where spatial variation of DQ information is shared by all coverages within a library, and DQ attributes are well defined. Recognizing that some nonstandard DQ information may be associated with certain regions, simple relational tables with fields keyed to coverage identifiers can be constructed (FIGURE 49). Line feature attributes may also be stored in a separate table.



DQ.AFT				
Id	Fac_id	Date	Source	Reliability
1	2	6/78	ONC	Good
2	3	7/83	ONC	Fair

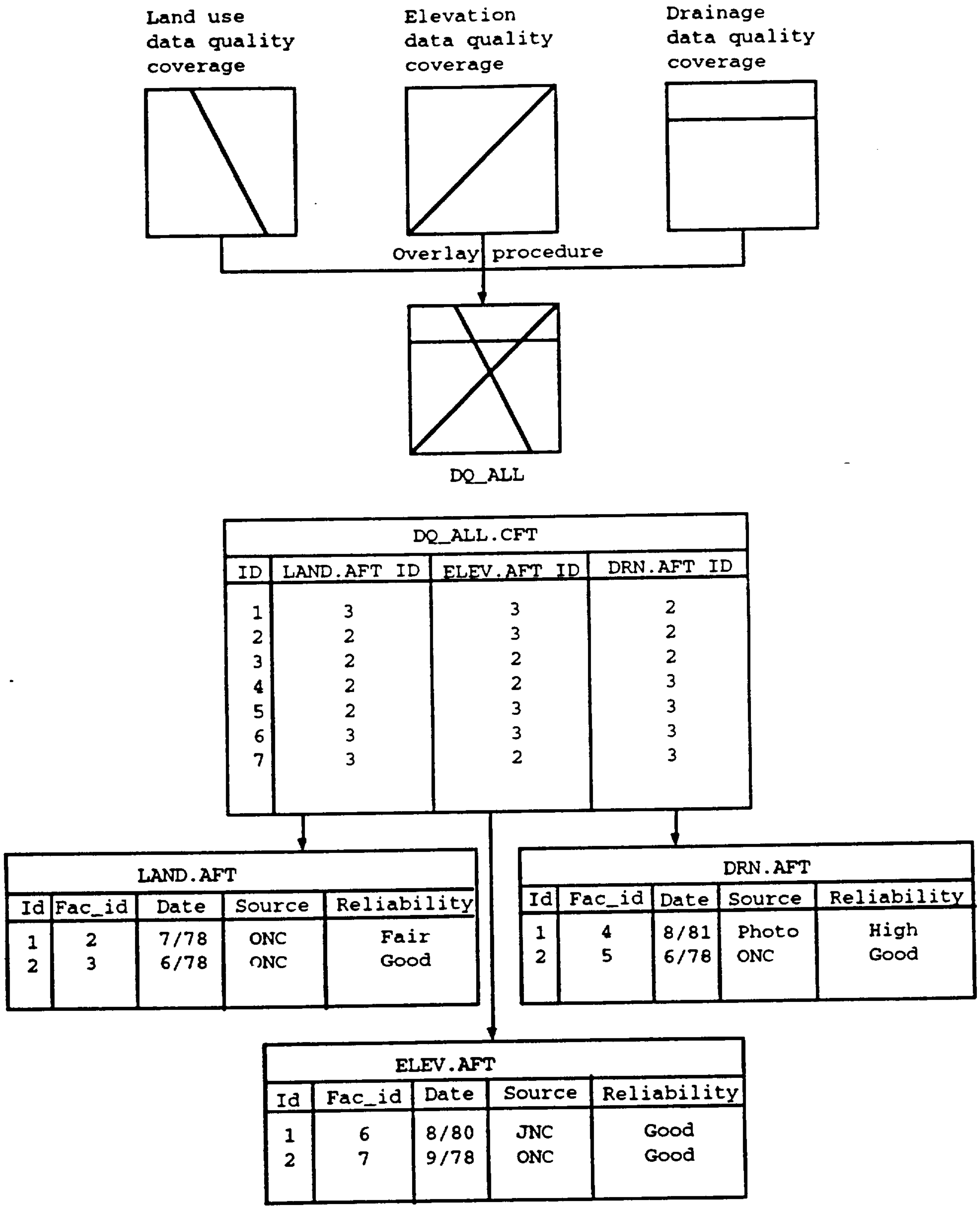
DQ.COM			
Id	DQ.AFT_ID	Coverage	Comments
1	2	HYNET	Some contours are missing

Related table for other coverage comments

FIGURE 49. Data quality coverage design-1.

E.4.4.2.2 Coverage-specific information. Section E.4.4.2.1 describes the basic constructs for describing characteristics of any data set with common spatial components to the reliability information. However, in some cases, the data producer may wish to describe characteristics within a single coverage, where quality information has spatial extents that vary from coverage to coverage. FIGURE 50 describes a scenario for organizing information under these conditions. The basic approach is to develop a single integrated coverage where the smallest faces are the product of the intersection of the various coverage-based data. Coverages organized in this manner are relatively easy to maintain, particularly for selective updating where new faces affect more than one primary data coverage. An alternative approach would be to maintain a series of separate coverages.

E.4.5 Conclusions. VPF provides a number of options for encoding data quality information. The information itself can be encoded at any level within the VPF structure depending upon its basic thematic and spatial characteristics. VPF data producers are encouraged to make use of the data quality table whenever possible. In instances where DQ characteristics vary spatially, the use of data quality coverages is strongly recommended.


Land use data quality coverage
Elevation data quality coverage
Drainage data quality coverage
Overlay procedure
DQ_ALL

DQ_ALL.CFT

ID	LAND.AFT ID	ELEV.AFT ID	DRN.AFT ID
1	3	3	2
2	2	3	2
3	2	2	2
4	2	2	3
5	2	3	3
6	3	3	3
7	3	2	3

LAND.AFT

Id	Fac_id	Date	Source	Reliability
1	2	7/78	ONC	Fair
2	3	6/78	ONC	Good

DRN.AFT

Id	Fac_id	Date	Source	Reliability
1	4	8/81	Photo	High
2	5	6/78	ONC	Good

ELEV.AFT

Id	Fac_id	Date	Source	Reliability
1	6	8/80	JNC	Good
2	7	9/78	ONC	Good

FIGURE 50. <u>Data quality coverage design-2</u>.

APPENDIX F

SPATIAL INDEXING

F.1. GENERAL

F.1.1 Scope. This appendix provides information and discussion concerning spatial indexes in a VPF database. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

F.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

F.3. DEFINITIONS

For purposes of this appendix, the definitions of section 3 of the main document shall apply.

F.4. GENERAL INFORMATION

F.4.1 Introduction. Spatial queries are queries in which the user points at a specific position on a display device containing a graphic representation of the data and asks (for example) "What is this line?" In order to answer the spatial query, any software that conducts a spatial query on a VPF database must search the edge primitive table for an exact match with "this line." Without a spatial index, the software would have to search every vertex of every edge sequentially for the correct response.

The purpose of a spatial index is to improve the speed with which software can retrieve a specific set of row ids from a primitive table. If the database contains spatial indexes, the software, when given a spatial query like, "What are the features within this bounding region?" can quickly retrieve the primitives that match the query. For each primitive (face, edge, entity node, connected node, and text), there can exist a spatial index file: FSI, ESI, NSI, CSI, or TSI (see section 5.4.2).

F.4.2 Categories of spatial decomposition. The spatial index is the second of four categories of spatial decomposition of a VPF database. The other three are the tile directory, the minimum bounding rectangle of the edge and face primitives, and the primitive coordinates. All four categories of spatial decomposition are described below.

F.4.2.1 Tile directory. Tiles in an implementation of VPF maintain spatially distributed primitives in separate directories. Thus, software developed for a tiled VPF database can search for data in only the relevant tile after the appropriate tile has been identified.

F.4.2.2 Spatial index. The second step in a typical software query is to use the appropriate index file (if one has been created within the database design). It is recommended that spatial index files associated with the primitives be created for every product implementation of VPF. Spatial indexes are discussed further below.

F.4.2.3 Minimum bounding rectangle (MBR). VPF requires that face and edge primitives have associated bounding rectangle table files–FBR and EBR. These tables allow the rapid retrieval of the primitives' spatial extent and are used by the software after the spatial index routine generates the primitive ids for the current spatial query. The bounding rectangle coordinates are typically used by the software to check the validity of the primitive ids in satisfying the query.

F.4.2.4 Primitive coordinates. It is necessary for software to exhaustively check nodes and text primitives for satisfaction of a spatial query, since these primitives do not have associated minimum bounding rectangles. The coordinate of the primitive is thus used to ensure the accurate retrieval of primitive ids output from the spatial index.

F.4.3 VPF spatial index file. The spatial index file internal structure in VPF is based on an adaptive grid binary tree. This method is powerful because it can handle all types of spatial queries (point, line, and area). The input primitives are broken down into a grid-based binary tree. At each cell (of the tree) there is a list of primitive MBRs and a list of the primitive ids that are found at this level of the tree.

The tree is created by storing primitive ids at a cell of the tree. "Bucket size" is the number used to determine when to split a cell and is defined by the product specification. A typical bucket size is eight (8). If the cell fills to the bucket size, then the cell is split into right and left children of the cell. The primitive ids are then distributed down into either child depending on the primitive MBRs. Only if a primitive MBR intersects the boundary between a parent's children cells, will the primitive id remain in the parent cell. Since primitives cannot be split between two cells, the number of primitives stored in a cell may exceed the bucket size. The process of splitting cells and distributng primitives based on each primitive's mbr and the bucket size continues until no cell requires splitting or the subdivision process reaches 255, the coordinate integer limit (see F.4.3.2). Product Specifications may limit the number cells in the grid-based binary tree for performance reasons (see F.4.4.1)

When examined spatially, the spatial index divides a tile into subelements (the cells of the tree); FIGURE 51. Each split results in dividing the parent cell into half. The first split is into right and left halves; the next split is into top and bottom halves; then right and left halves again; and so on.

The actual format of the spatial index file consists of the following:

a. A header containing the number of primitives, the minimum bounding rectangle of the entire spatial extent of the tree, and the number of cells in the tree. Note that the MBR of the entire spatial extent of the tree will coincide with the tile boundary in level three coverages for face spatial indices (fsi) edge spatial indices (esi), and connected node spatial indices (csi). However, there is no reason to force the tile boundary as the spatial extent of the grid-based binary tree for entity node spatial indices (nsi) or for coverages with topology levels less than three. In fact, using the primitives MBR as the spatial extent of the tree rather than the tile boundary provides for a more efficient spatial index.

b. A bin array of the tree. Each bin contains two items. A beginning location (offset from the end of the BIN Array Record) for the cell's data and the number of primitives in the cell. All intermediate cells are listed, even if empty. The offset for an empty cell equals zero. The last entry in the bin array record is always a populated cell. The final level of the tree is not forced to be balanced by creating empty cells.

c. Data records for each primitive in the tree. There is one record for each primitive in the tree. Each record contains four 1-byte integers defining the MBR for a primitive and that primitive's ID.

F.4.3.1 <u>Tree navigation</u>. For any cell, the cell from which it was generated is the integer value obtained by dividing by two. Thus, cell 3 points back to cell 1 [INT(3/2)], as does cell 2.

New cells created by splitting are numbered by multiplying the current cell by two and adding one for the second new cell. Thus, cell 2 becomes cells 4 and 5, and cell 3 becomes cells 6 and 7.

F.4.3.2 Spatial index coordinate system. The coordinate system for the spatial index is based upon 1-byte integers, so a primitive's MBR must be converted to the spatial index coordinate system. All coordinates are relative to the lower left corner of the tile and range from 0 to 255.

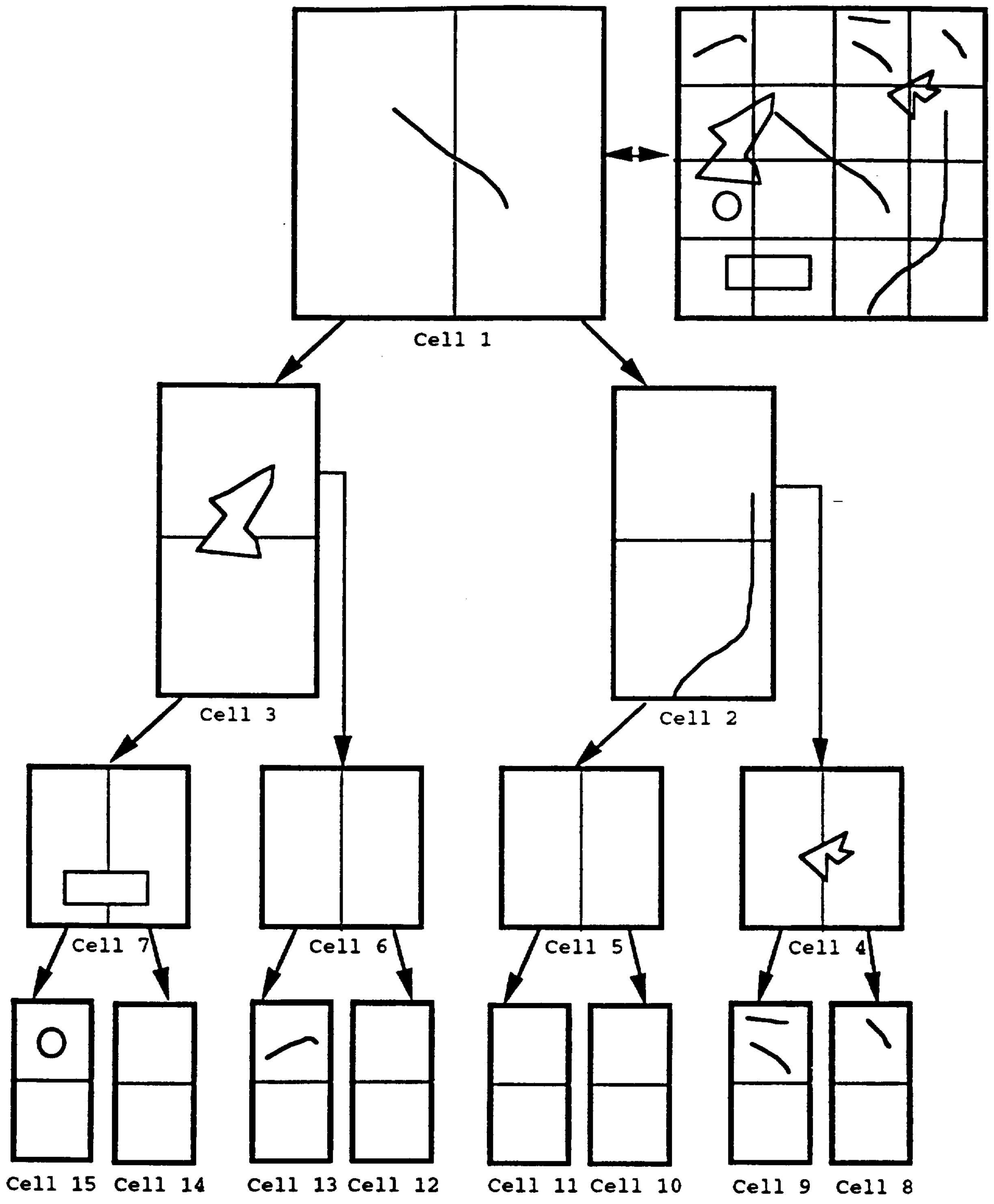


FIGURE 51. Spatial index cell decomposition.

F.4.4 Examples of spatial index creation. TABLE 69 is a listing of the minimum (x_1, y_1) and maximum (x_2, y_2) coordinates of the MBRs of 19 face primitives. The coordinate values in TABLE 66

TABLE 69. Minimum and maximum coordinates for 19 primitives in a tile. Universe is primitive number 1.

Primitive ids	x_1 (deg)	y_1 (deg)	x_2 (deg)	y_2 (deg)
1	Null	Null	Null	Null
2	-5.00	54.63	-3.57	55.00
3	-5.00	52.00	-2.74	55.00
4	-5.00	54.91	-4.99	54.94
5	-5.00	54.76	-4.99	54.77
6	-4.80	54.06	-4.31	54.42
7	-3.28	54.05	-3.17	54.15
8	-0.71	53.54	0	53.74
9	-0.57	53.68	-0.53	53.69
10	-4.60	53.13	-4.05	53.43
11	-4.71	53.24	-4.56	53.33
12	-4.80	52.75	-4.78	52.77
13	-5.00	50.53	-2.35	51.82
14	-4.71	51.63	-4.68	51.65
15	-4.68	51.16	-4.65	51.20
16	-1.03	50.78	-0.95	50.84
17	-1.59	50.58	-1.08	50.77
18	-1.99	50.69	-1.96	50.70
19	-5.00	50.16	-4.99	50.17

are all in degrees. The primitives are all located within a 5- by 5-degree tile that has an MBR of (-5, 50), (0, 55). The MBR coordinates can be converted to the spatial index coordinate system by using the following equations:

(1) for minimum (x_1, y_1)

$$\text{integer truncation of} \left[\frac{\text{(Coordinate - minimum)}}{\text{(Maximum - minimum)}} \text{ X } 255 \right]$$

(2) for maximum (x_2, y_2)

$$\left[\text{integer truncation of} \left[\frac{\text{(Coordinate - minimum)}}{\text{(Maximum - minimum)}} \text{ X } 255 \right] \right] +1$$

If equation (2) results in a value of 256, the value must be set to 255.

Our example thus becomes:

$$y_1 = \frac{(\text{Latitude} - 50)}{(55 - 50)} \times 255$$

and

$$x_1 = \frac{(\text{Longitude} + 5)}{(0 + 5)} \times 255$$

The results of this conversion are listed in TABLE 70.

TABLE 70. Minimum and maximum spatial index coordinates.

Primitive ids	x_1	y_1	x_2	y_2
2	0	236	74	255
3	0	102	116	255
4	0	250	1	252
5	0	242	1	244
6	10	207	36	226
7	87	206	94	212
8	218	180	255	191
9	225	187	228	189
10	20	159	49	175
11	14	165	23	170
12	9	140	12	142
13	0	27	136	93
14	14	83	17	85
15	16	59	18	62
16	202	39	207	43
17	173	29	200	40
18	153	35	155	36
19	0	8	1	9

NOTE: Since face 1 (universe face) is a topological artifact (i.e. no outer ring), the MBR in normalized coordinates cannot be calculated. Therefore face 1 is not included in the bin data record portion of the index. Further, no fsi is built for a face table containing only the universe face.

If the MBRs of each primitive are plotted, they appear as shown in FIGURE 52. In FIGURE 53, dividing lines have been added to FIGURE 52 to show that primitive 13 is present in both the left and right halves of the tile, and that primitive 3 is present in both the top and bottom halves of the tile.

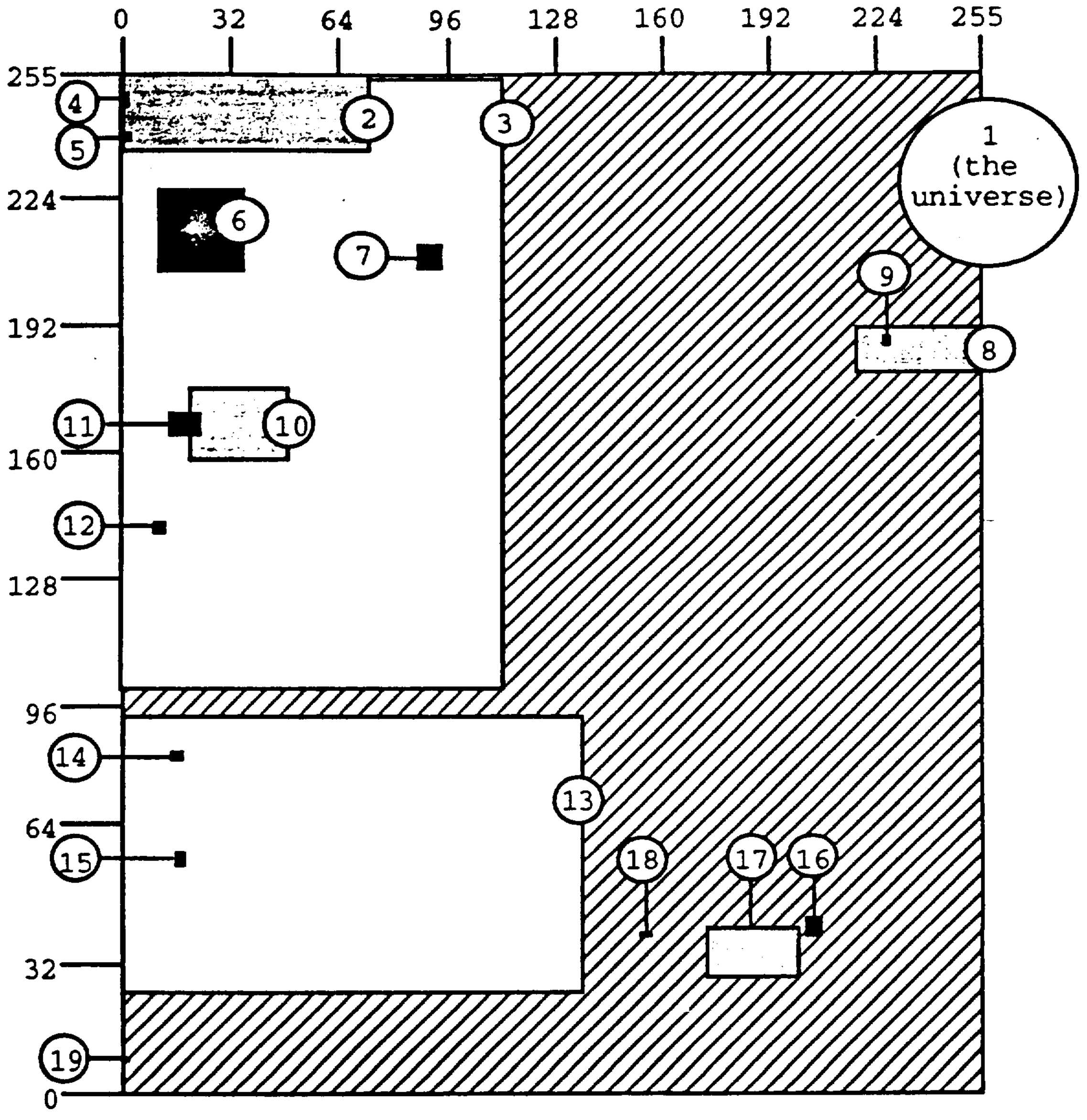


FIGURE 52. Location of MBRs in tile.

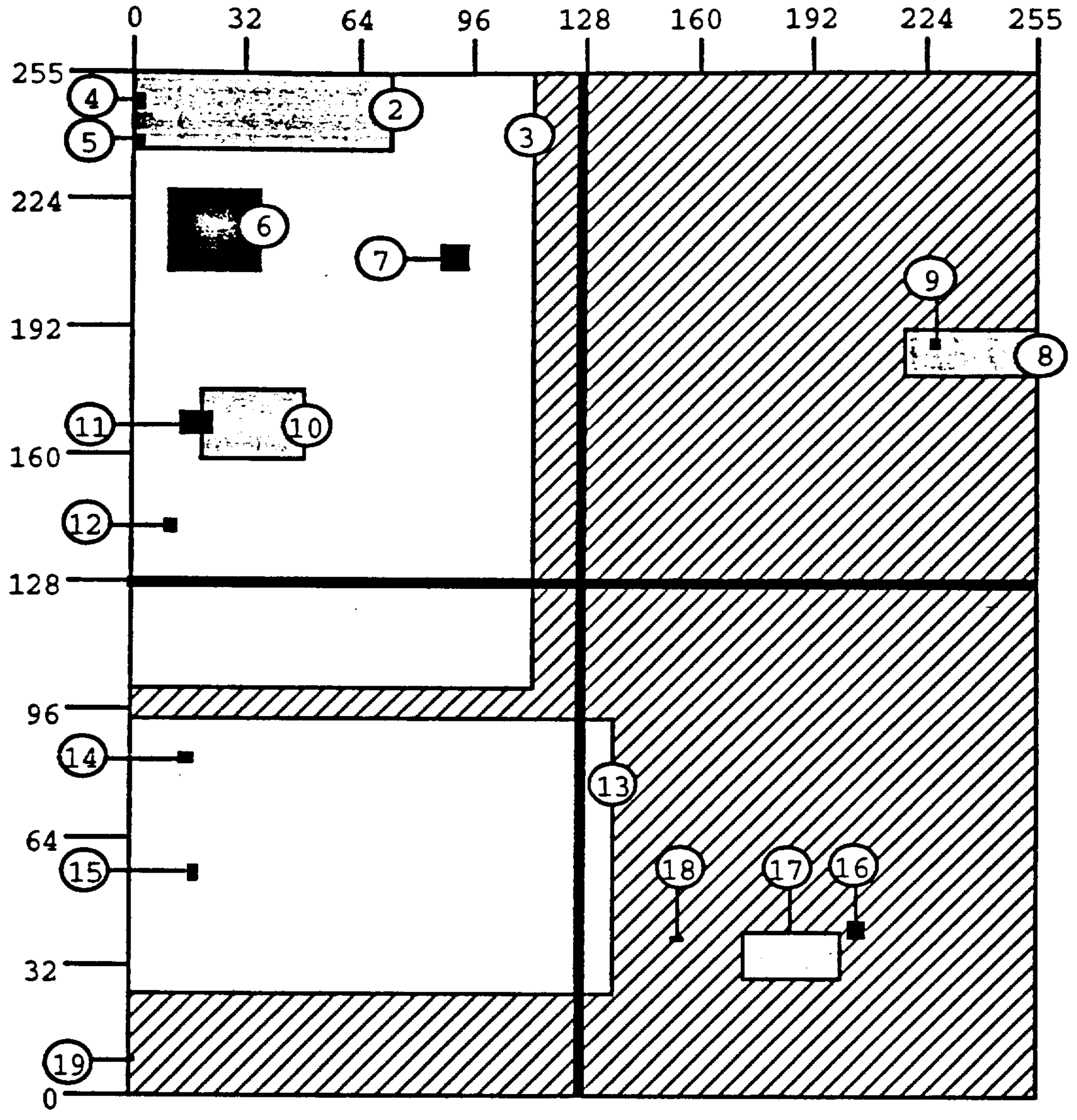


FIGURE 53. Tile content divided in four quarters.

F.4.4.1 Example of tree creation. The bucket size for this example is set at eight. If a parent cell contains nine or more primitives that can be propagated to the next level, the parent splits into two children. The first split of the tile puts primitives 8, 9, 16, 17, and 18 into cell 2 and places primitive 3 into cell 3 (FIGURE 54). Primitive 13 must be held in cell 1 (FIGURE 55) because neither of the split cells contains the entire MBR for primitive 13.

Cell 3 (before the split into cells 6 and 7) contains twelve primitives: 2, 3, 4, 5, 6, 7, 10, 11, 12, 14, 15, and 19. Since

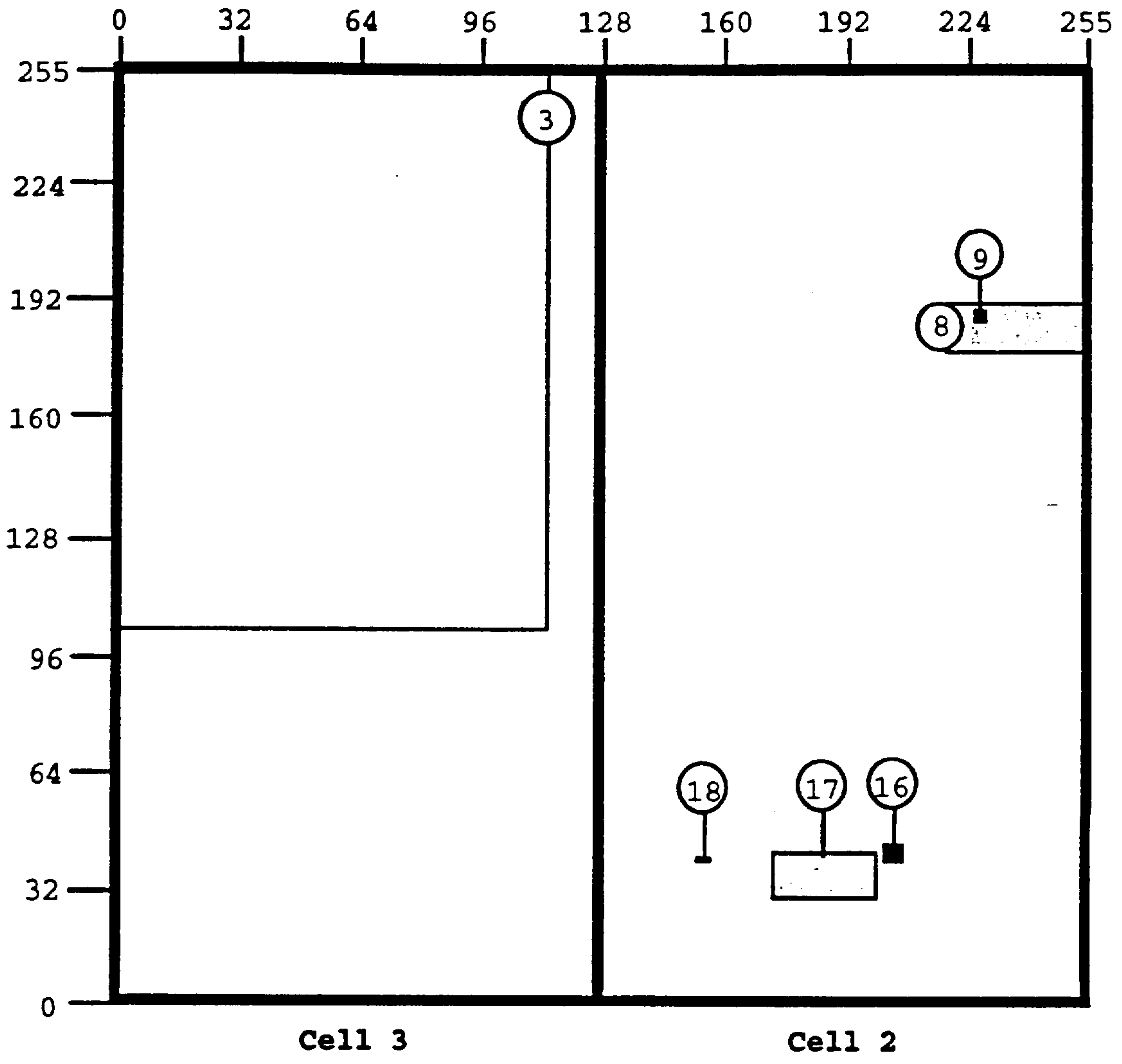
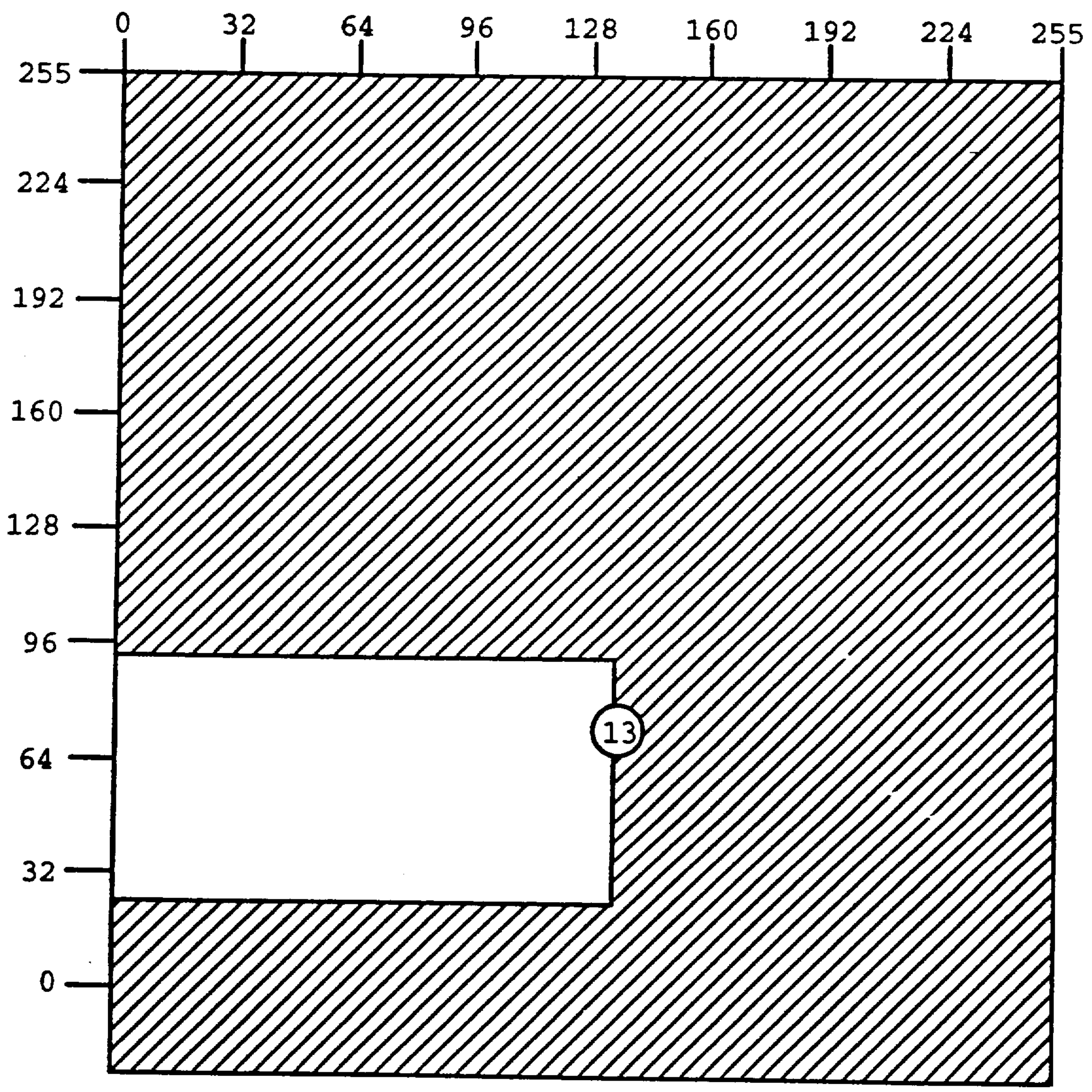


FIGURE 54. First cell split.

this exceeds the defined bucket size, cell 3 is split (FIGURE 54). The MBR of primitive 3 will cross the boundary of cells 6 and 7. Therefore it must remain in cell 3. Eleven primitives remain. Based on each primitive's MBR, the contents of the new cells are: primitives 2, 4, 5, 6, 7, 10, 11, and 12 in cell 6 and primitives 14, 15, and 19 in cell 7. Note that no primitives are allocated to cells 4 and 5, since all five primitives on the right half of the tile could be held in cell 2. The five primitives do not 'over-flow' the defined bucket size. All of the primitives in this example have now been allocated to the tree.

In theory, the process of splitting cells and distributng primitives based on each primitive's mbr and the bucket size continues until no cell requires splitting or the subdivision process reaches the bottom level of the grid-based binary tree (where each cell is 1x1 in normalized coordinates). This implies a grid-based binary tree with 17 levels and 131,071 cells (2^{17} - 1). As noted in E.4.3, however, Product Specifications may limit the number of cells allowed in the grid-based binary tree for performance reasons.

The spatial index that results for this example is shown in TABLE 71.



Cell 1
FIGURE 55. Content of cell 1.

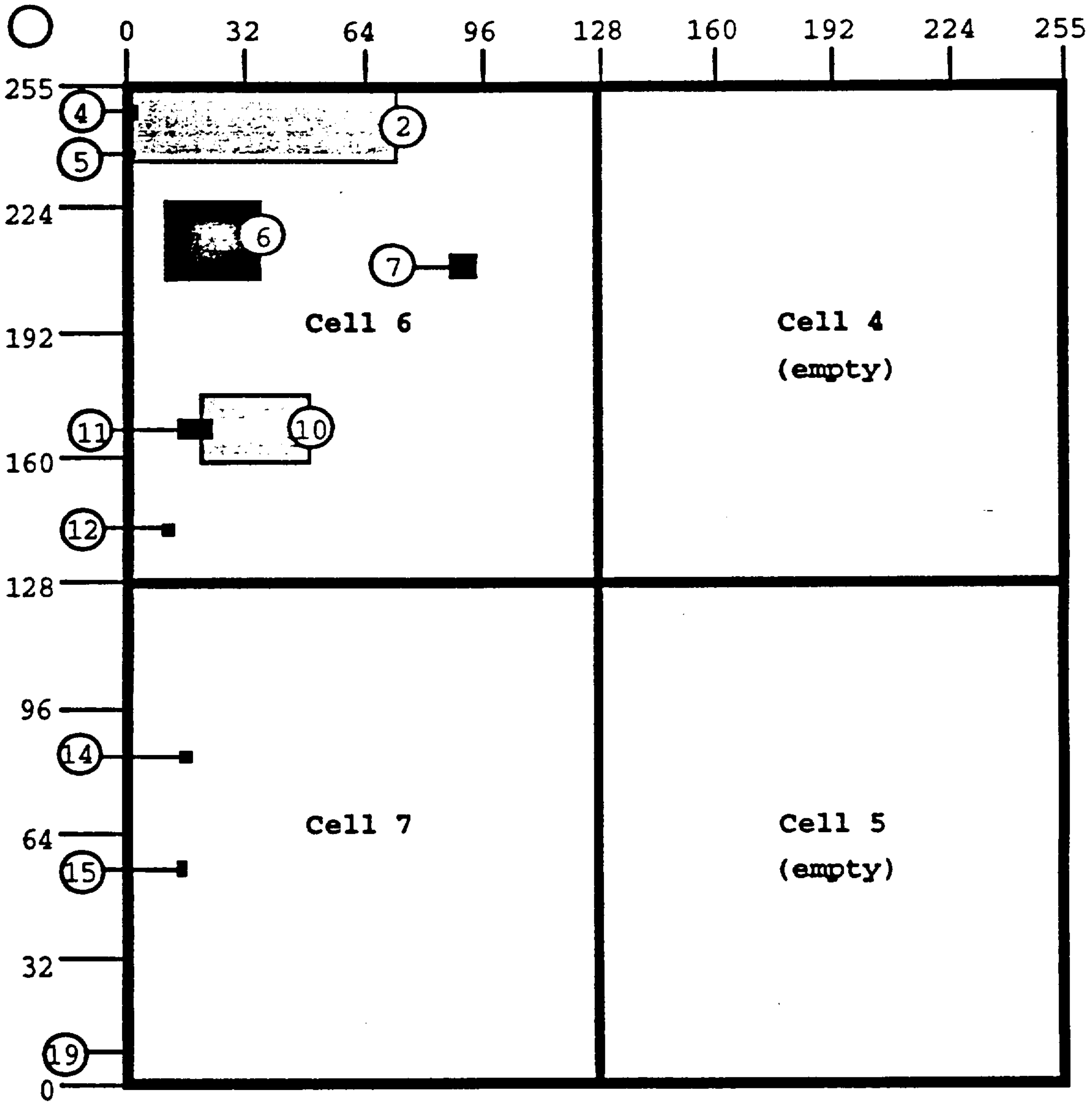


FIGURE 56. Second cell split.

TABLE 71. Example of spatial index.

Header:

Number of primitives: 18
MBR: -5.00, 50.00, 0.00, 55.00
Number of cells: 7

Bin Array Record:

Cell (information not in actual spatial index)	Offset	Primitive count
1	0	1
2	8	5
3	48	1
4	0	0
5	0	0
6	56	8
7	120	3

Bin Data Record:

Record Number	Offset Address	x_1	y_1	x_2	y_2	Primitive ids
Cell 1						
1	0	0	26	136	94	13
Cell 2						
2	8	153	35	155	36	18
3	16	173	29	200	40	17
4	24	202	39	207	43	16
5	32	226	187	228	189	9
6	40	218	180	255	191	8
Cell 3						
7	48	0	102	116	255	3
Cell 6						
8	56	87	206	94	212	7
9	64	10	206	36	226	6
10	72	0	242	1	244	5
11	80	0	250	1	253	4
12	88	0	236	74	255	2
13	96	20	159	49	176	10
14	104	14	165	23	170	11
15	112	9	140	12	142	12
Cell 7						
16	120	0	8	1	9	19
17	128	16	59	18	62	15
18	136	14	83	17	85	14

F.4.5 Spatial query. A spatial index can support a spatial query in several ways. For node primitives, software may require the point to be within a specified distance of a pixel designated during the query process, or within a box generated during the query process. For an edge primitive, the software may specify that candidate edges are to be within some specified perpendicular distance of the query pixel or have MBRs that intersect a query box. For a face primitive, the software may define the candidate edges as having MBRs that intersect a query box, be fully contained within the query MBR, or be determined by solving the "point-in-polygon" puzzle. In any case, the spatial index forms the starting point for the database search. It works as follows:

a. The user designates a query point (pixel).

b. The software converts the query point into the spatial index coordinate system ((0,0) to (255,255)).

c. The software tests all features within the top level cell, if any, to determine if these features qualify for the query response.

d. The software calculates the next smaller cell containing the query point.

e. The software repeats the test (if necessary) and continues to decrease cell size until no more records exist.

F.4.6 Spatial query using the sample tree.

a. The user designates a query point and the software determines the normalized coordinates at (192, 32).

b. Beginning at the root of the tree, the software goes to cell 1 and finds face 13.

c. The software checks the MBR of face 13 to determine if it includes the query point.

d. Since face 13 does not include the query point the software calculates the children of cell 1, which are cells 2 and 3. Cell 3 cannot contain the query point, so cell 2 is examined. Faces 8, 9, 16, 17, and 18 are found.

e. The software checks the MBR of these faces to determine if they include the query point.

f. The software reports the query result which is face 17.

APPENDIX G

CODING FOR METADATA TABLES

G.1. GENERAL

G.1.1 Scope. The following codes are used in the VPF metadata tables found at the library and database levels. In particular, these coding schemes must be used in the geographic reference table (TABLE 37) to ensure the proper interpretation of the coordinate system in a VPF library.

G.2. APPLICABLE DOCUMENTS

This section is not applicable to this appendix.

G.3. CODING SCHEMES

TABLE 72. Vertical datum codes.

Code	Description
000	Unknown
002	High water
003	Higher high water
004	Indian spring low water
005	Low water
006	Lower low water
007	Mean high water
008	Mean high water neaps
009	Mean high water springs
010	Mean higher high water
011	Mean low water
012	Mean low water neaps
013	Mean low water springs
014	Mean lower low water
015	Mean sea level
016	Mean tide level
017	Neap tide
018	Spring tide
019	Mean lower low water springs
020	Lowest astronomical tide
021	Chart datum
022	Highest astronomical tide
999	Other

TABLE 73. Coding for units of measure.

Code	Description
CM	Centimeters
DEG	Degrees of Arc
DM	Decimeters
FF	Fathoms and Feet
FM	Fathoms
FT	Feet
IN	Inches
KM	Kilometers
M	Meters
MA	Minutes of Arc
MI	Statute Miles
ML	Mils
MM	Millimeters
NM	Nautical Miles
SEC	Seconds of Arc
UM	Micrometers
YD	Yards

TABLE 74. Coding for ellipsoids.

	Ellipsoid	Code
1	Modified Airy	AAM
2	Airy	AAY
3	Australian National	AUN
4	Bessel 1841	BES
5	Clarke 1858	CLE
6	Clarke 1880	CLJ
7	Clarke 1866	CLK
8	Everest	EVE
9	Modified Everest	EVM
10	Modified Fischer 1960	FAM
11	Fischer	FIS
12	Geodetic Reference System 1967	GRE
13	Geodetic Reference System 1980	GRS
14	Helmert 1906	HEL
15	Hough	HOU
16	Indonesian 1974	IDN
17	International	INT
18	Krassovsky	KRA
19	South American 1969	SAM
20	Walbeck	WAL

21	World Geodetic System 1960	WGA
22	World Geodetic System 1966	WGB
23	World Geodetic System 1972	WGC
24	World Geodetic System 1984	WGE

TABLE 74. Coding for ellipsoids - Continued.

TABLE 75. Coding for geodetic datums.

	Geodetic Datums	Code
1	Adindan	ADI
2	Adindan (Ethiopia)	ADIA
3	Adindan (Sudan)	ADIB
4	Adindan (Mali)	ADIC
5	Adindan (Senegal)	ADID
6	Adindan (Mean value: Ethiopia and Sudan)	ADIM
7	Afgooye (Somalia)	AFG
8	Ain el Abd 1970 (Bahrain Island)	AIN
9	Anna 1 Astro (Cocos Islands)	ANO
10	Arc 1950	ARF
11	Arc 1950 (Botswana)	ARFA
12	Arc 1950 (Lesotho)	ARFB
13	Arc 1950 (Malawi)	ARFC
14	Arc 1950 (Swaziland)	ARFD
15	Arc 1950 (Zaire)	ARFE
16	Arc 1950 (Zambia)	ARFF
17	Arc 1950 (Zimbabwe)	ARFG
18	Arc 1950 (Mean value: Botswana, Lesotho, Malawi, Swaziland, Zaire, Zambia, and Zimbabwe)	ARFM
19	Arc 1960 (Kenya)	ARSA
20	Arc 1960 (Tanzania)	ARSB
21	Arc 1960 (Mean value: Kenya, Tanzania)	ARSM
22	Ascension Island 1958 (Ascension Island)	ASC
23	Astro Station 1952 (Marcus Island)	ASQ
24	Astro Beacon "E" (Iwo Jima Island)	ATF
25	Average Terrestrial System (Atlantic Datum) 1997	ATS
26	Australian Geod. 1966 (Australia and Tasmania Is.)	AUA
27	Australian Geod. 1984 (Australia and Tasmania Is.)	AUG
28	Djakarta (Batavia) (Sumatra Island, Indonesia)	BAT
29	Bermuda 1957 (Bermuda Islands)	BER
30	Bogota Observatory (Colombia)	BOO
31	Bukit Rimpah (Bangka & Belitum Islands, Indonesia)	BOR
32	Bukit Rimpah	BUR
33	Canton Astro 1966 (Phoenix Islands)	CA0

34	Cape Canaveral (Mean value: Florida and Bahama Islands)	CAC
35	Campo Inchauspe (Argentina)	CAI
36	Cape (South Africa)	CAP
37	Camp Area Astro (Camp McMurdo Area, Antartica)	CAZ
38	Carthage (Tunisia)	CGE
39	Chua Astro (Paraguay)	CHG
40	Chatham 1971 (Chatham Island, New Zealand)	CHI
41	Chua Astro	CHU
42	Corrego Alegre (Brazil)	COA
43	Guyana CSG67	CSG
44	GUX 1 Astro (Guadacanal Island)	DOB
45	Easter Island 1967 (Easter Island)	EAS
46	European 79	ENB
47	Wake-Eniwetok 1960 (Marshall Islands)	ENW
48	European 1979 (Mean value: Austria, Finland, Netherlands, Norway, Spain, Sweden, and Switzerland)	EUQ
49	European 1950 (Mean value)	EUR
50	European 1950 (Western Europe: Austria, Denmark, France, Federal Republic of Germany , Netherlands, and Switzerland)	EURA
51	European 1950 (Greece)	EURB
52	European 1950 (Norway and Finland)	EURC
53	European 1950 (Portugal and Spain)	EURD
54	European 1950 (Cyprus)	EURE
55	European 1950 (Egypt)	EURF
56	European 1950 (Iran)	EURH
57	European 1950 (Sardinia)	EURI
58	European 1950 (Sicily)	EURJ
59	European 1950 (England, Channel Islands, Ireland, Northern Ireland, Scotland, Shetland Islands, and Wales)	EURK
60	European 1950 (Mean value: Austria, Belgium, Denmark, Finland, France, Federal Republic of Germany, Gibraltar, Greece, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, & Switzerland)	EURM
61	Oman (Oman)	FAH
61a	French NTF	FDA
62	Observatorio 1966 (Corvo and Flores Islands, Azores)	FLO
63	GAN Datum (Addu Atoll, Republic of Maldives)	GAA
64	German	GDA
65	Geodetic Datum 1949 (New Zealand)	GEO
66	Ghana	GHA

TABLE 75. Coding for geodetic datums - Continued.

67	DOS 1968 0230 (Gizo Island, New Georgia Islands)	GIZ
68	SW Base (Faial, Graciosa, Pico, Sao Jorge, and Terceira Island, Azores)	GRA
69	Genung Segara (Kalimantan Island, Indonesia)	GSE
70	G. Serindung	GSF
71	Guam 1963	GUA
72	Guadeloupe Ste. Anne	GUD
73	Herat North (Afganistan)	HEN
74	Hermannskogel	HER
75	Prov. S. Chilea.i (S. Chile, 53 S.)	HIT
76	Hjorsey 1955 (Iceland)	HJO
77	Hong Kong 1963 (Hong Kong)	HKD
78	Hu-tzu-shan	HTN
79	Bellevue (IGN) (Efate and Erromango Islands)	IBE
80	Italian	IDA
81	Indian	IND
82	Indian (Thailand and Vietnam)	INDA
83	Indian (Bangladesh, India, and Nepal)	INDB
84	Ireland 1965	IRE
85	Ireland 1965 (Ireland and Northern Ireland)	IRL
86	ISTS 073 Astro 1969 (Diego Garcia)	IST
87	Johnston Island 1961 (Johnston Island)	JOH
88	Kandawala (Sri Lanka)	KAN
89	Kertau 1948 (West Malaysia and Singapore)	KEA
90	Kerguelen Island 1949 (Kerguelen Island)	KEG
91	L.C. 5 Astro 1961 (Cayman Brac Island)	LCF
92	Liberia 1964 (Liberia)	LIB
93	Local Astro.	LOC
94	Luzon	LUZ
95	Luzon (Philipines except Mindanao Island)	LUZA
96	Luzon (Mindanao Island)	LUZB
97	Martinique Fort-Desaix	MAR
98	Marco Astro (Salvage Islands)	MAA
99	Massawa (Eritrea, Ethiopia)	MAS
100	Mayotte Combani	MAY
101	Merchich	MER
102	Merchich (Morocco)	MER
103	Midway Astro 1961 (Midway Island)	MID
104	Mahe 1971 (Mahe Island)	MIK
105	Minna (Nigeria)	MIN
106	Rome 1940 (Sardinia Island)	MOD
107	Montjong Lowe	MOL

TABLE 75. Coding for geodetic datums - Continued.

108	Viti Levu 1916 (Viti Levu Island, Fiji Islands)	MVS
109	Nahrwan (Masirah Island, Oman)	NAHA
110	Nahrwan (United Arab Emirates)	NAHB
111	Nahrwan (Saudi Arabia)	NAHC
112	Naparima (BWI Trinidad and Tobago)	NAP
113	North American 1983 (Mean Value: Alaska, Canada, CONUS, Mexico, and Central America)	NAR
114	North American 1927 (Mean value)	NAS
115	North American 1927 (Eastern US)	NASA
116	North American 1927 (Western US)	NASB
117	North American 1927 (Mean value: CONUS)	NASC
118	North American 1927 (Alaska)	NASD
119	North American 1927 (Mean value: Canada)	NASE
120	North American 1927 (Alberta and British Columbia)	NASF
121	North American 1927 (Newfoundland, New Brunswick, Nova Scotia and Quebec)	NASG
122	North American 1927 (Manitoba and Ontario)	NASH
123	North American 1927 (Northwest Territories and Saskatchewan)	NASI
124	North American 1927 (Yukon)	NASJ
125	North American 1927 (Mexico)	NASL
126	North American 1927 (Central America - Belize, Costa Rica, El Salvador, Guatemala, Honduras, and Nicaragua)	NASN
127	North American 1927 (Canal Zone)	NASO
128	North American 1927 (Caribbean, Barbados, Caicos Islands, Cuba, Dominican Republic, Grand Cayman, Jamaica, Leeward Islands, and Turks Islands)	NASP
129	North American 1927 (Bahamas, except San Salvador Island)	NASQ
130	North American 1927 (San Salvador Island)	NASR
131	North American 1927 (Cuba)	NAST
132	North American 1927 (Hayes Peninsula, Greenland)	NASU
133	North American 1983	NAX
134	Nigeria	NIG
135	Old Egyptian (Egypt)	OEG
136	Ordnance Survey of Great Britain	OGB
137	Ord. Survey G.B. 1936 (England)	OGBA
138	Ord. Survey G.B. 1936 (England, Isle of Man, and Wales)	OGBB
139	Ord. Survey G.B. 1936 (Scotland and Shetland Islands)	OGBC
140	Ord. Survey G.B. 1936 (Wales)	OGBD
141	Ord. Survey G.B. 1936 (Mean value: England, Isle of Man, Scotland, Shetland, and Wales)	OGBM

TABLE 75. Coding for geodetic datums - Continued.

142	Old Hawaiian	OHA
143	Old Hawaiian (Hawaii)	OHAA
144	Old Hawaiian (Kauai)	OHAB
145	Old Hawaiian (Maui)	OHAC
146	Old Hawaiian (Oahu)	OHAD
147	Old Hawaiian (Mean value)	OHAM
148	Pitcairn Astro 1967 (Pitcairn Island)	PIT
149	Pico de las Nieves (Canary Islands)	PLN
150	SE Base (Porto Santo) (Porto Santo & Madeira Islands)	POS
151	Provisional South American 1956	PRP
152	Prov. S. Amer. 1956 (Northern Chile near 19 degrees south)	PRPA
153	Prov. S. Amer. 1956 (Southern Chile near 43 degrees south)	PRPC
154	Prov. S. Amer. 1956 (Columbia)	PRPD
155	Prov. S. Amer. 1956 (Ecuador)	PRPE
156	Prov. S. Amer. 1956 (Guyana)	PRPF
157	Prov. S. Amer. 1956 (Peru)	PRPG
158	Prov. S. Amer. 1956 (Venezuela)	PRPH
159	Prov. S. Amer. 1956 (Mean value: Bolivia, Chile, Colombia, Ecuador, Guyana, Peru, & Venezuela)	PRPM
160	Puerto Rico (Puerto Rico and Virgin Islands)	PUR
161	Qatar National (Qatar)	QAT
162	Qornoq (South Greenland)	QUO
163	Reunion 1947	REU
164	Santo (DOS) 1965 (Espirito Santo Island)	SAE
165	South American 1969 (Argentina)	SANA
166	South American 1969 (Bolivia)	SANB
167	South American 1969 (Brazil)	SANC
168	South American 1969 (Chile)	SAND
169	South American 1969 (Columbia)	SANE
170	South American 1969 (Ecuador)	SANF
171	South American 1969 (Guyana)	SANG
172	South American 1969 (Paraguay)	SANH
173	South American 1969 (Peru)	SANI
174	South American 1969 (Trinidad and Tobago)	SANK
175	South American 1969 (Venezuela)	SANL
176	South American 1969 (Mean value: Argentina, Bolivia, Brazil, Chile, Columbia, Ecuador, Guyana, Paraguay, Peru, Trinidad and Tobago, and Venezuela)	SANM
177	Sao Braz (Sao Miguel, Santa Maria Islands, Azores)	SAO
178	Sapper Hill 1943 (East Falkland Islands)	SAP
179	Schwarzeck (Namibia)	SCK

TABLE 75. Coding for geodetic datums - Continued.

180	Astro Dos 71/4 (St. Helena Island)	SHB
181	Sierra Leone 1960	SIB
182	South Asia (Southeast Asia, Singapore)	SOA
183	St. Pierre et Miquelon 50	STP
184	Tananarive Obsv. 1925	TAN
185	Tristan Astro 1968 (Tristan da Cunha)	TDC
186	Timbali 1948 (Brunei and East Malaysia - Sarawak and Sabah)	TIL
187	Tokyo (Mean value)	TOK
188	Tokyo (Japan)	TOYA
189	Tokyo (Korea)	TOYB
190	Tokyo (Okinawa)	TOYC
191	Tokyo (Mean value: Japan, Korea, and Okinawa)	TOYM
192	Astro Tern Is. 1961 (Tern Island, Hawaii)	TRN
193	Undetermined (processed as if WGS 84)	UND
194	Voirol	VOI
195	World Geodetic System 1960	WGA
196	World Geodetic System 1966	WGB
197	World Geodetic System 1972	WGC
198	World Geodetic System 1984	WGE
199	Yacare (Uruguay)	YAC
200	Zanderij (Surinam)	ZAN

TABLE 75. Coding for geodetic datums - Continued.

TABLE 76. Coding for projections.

NAME	CODE	PARAMETERS			
		1	2	3	4
Albers Equal Area	AC	Central Meridian	Std. Parallel Nearer to Equator	Std. Parallel Farther from Equator	Parallel of Origin
Azimuthal Equal Area	AK	Longitude of Tangency	Latitude of Tangency	-	-
Azimuthal Equal Distant	AL	Longitude of Tangency	Latitude of Tangency	-	-
Gnomonic	GN	Longitude of Tangency	Latitude of Tangency	-	-
Hotine Oblique Mercator (Rectified Skew Orthomorphic)	RB	Longitude of Proj. Origin	Latitude of Proj. Origin	Azimuth of Skew X-Axis at Proj. Origin	ScaleFactor at Proj. Origin
Lambert Conformal Conic	LE	Central Meridian	Std. Parallel Nearer to Equator	Std Parallel Farther from Equator	Parallel of Origin
Lambert Equal Area	LJ	Central Meridian	-	-	-

NAME	CODE	PARAMETERS			
		1	2	3	4
Mercator	MC	Central Meridian	Latitude of True Scale	Parallel of Origin	-
Oblique Mercator	OC	Longitude of Reference Point on Great Circle	Latitude of Reference Point on Great Circle	Azimuth of Great Circle at Reference Point	-
Orthographic	OD	Longitude of Tangency	Latitude of Tangency	-	-
Polar Stereo-graphic	PG	Central Meridian	Latitude of True Scale	-	-
Polyconic	PH	Central Meridian	-	-	-
Transverse Mercator	TC	Central Meridian	Central Scale Factor	Parallel of Origin	-
Oblique Stereographic	SD	Longitude of Origin	Latitude of Origin	Scale factor at Origin	-
Relative Coordinates	RC	X-Scale Factor	Y-Scale Factor		

TABLE 76. Coding for projections - Continued.

APPENDIX H

SAMPLE VPF DATABASE

H.1. SCOPE

1.1 Scope. This document provides a sample text description of a VPF database. The information contained in this standard shall be used by the Military Departments, Office of the Secretary of Defense, Organizations of the Joint Chiefs of Staff and the Defense Agencies of the Department of Defense (collectively known as DoD Components) in preparing and accessing digital geographic data required or specified to be in vector product format.

H.2. APLICABLE DOCUMENTS

This section is not applicable to this appendix.

H.3. EXAMPLES

The following pages document a sample VPF database. The general form includes a printout of the VPF header for each file, followed by the actual data. In some cases the data is truncated or formatted with more than one record per line, but in general it can be read by referring to header instructions. The data structure contains numerous levels of data with many files at each level. Selected files at various levels are provided in the following tables to illustrate the format and contents. No examples of indexes are provided.

The example tables are converted for ease of human interpretation by use of a software utility. Because of the conversion, the example tables are not completely VPF compliant.

TABLE 77. Database Header Table.

Table Definition:
Name: <ID> <Row Identifier> Type: I Count: 1 KeyType: P
Name: <VPF_VERSION> <VPF version number> Type: T Count: 10 KeyType: N
Name: <DATABASE_NAME> <Directory name of this database> Type: T Count: 8 KeyType: N
Name: <DATABASE_DESC> <Description of this database> Type: T Count: 100 KeyType: N
Name: <MEDIA_STANDARD> <Media Standard (ie: CDROM)> Type: T Count: 20 KeyType: N
Name: <ORIGINATOR> <Producer of this database> Type: T Count: 50 KeyType: N
Name: <ADDRESSEE> <Address of the producer> Type: T Count: 100 KeyType: N
Name: <MEDIA_VOLUMES> <Number of Volumes in this database> Type: T Count: 1
KeyType: N
Name: <SEQ_NUMBERS> <The Sequential Number(s) in this database> Type: T Count: 1
KeyType: N
Name: <NUM_DATA_SETS> <Number of Data Sets> Type: T Count: 1 KeyType: N
Name: <SECURITY_CLASS> <Security Classification> Type: T Count: 1 KeyType: N
Name: <DOWNGRADING> <Downgrading> Type: T Count: 3 KeyType: N
Name: <DOWNGRADE_DATE> <Date> Type: D Count: 1 KeyType: N
Name: <RELEASABILITY> <Releasability restrictions of data> Type: T Count: 20 KeyType: N
Name: <OTHER_STD_NAME> <Description of other data standards used> Type: T Count: 50
KeyType: N
Name: <OTHER_STD_DATE> <Date> Type: D Count: 1 KeyType: N
Name: <OTHER_STD_VER> <Version number of other standard> Type: T Count: 10
KeyType: N
Name: <TRANSMITTAL_ID> <Unique Transmittal Identifier> Type: T Count: 1 KeyType: N
Name: <EDITION_NUMBER> <Edition Number of this database> Type: T Count: 10
KeyType: N
Name: <EDITION_DATE> <Date of edition> Type: D Count: 1 KeyType: N

ID: 1
VPF_VERSION: TBD
DATABASE_NAME: DNC
DATABASE_DESC: Digital Nautical Chart database based on General, Harbor, Approach & Coastal charts
MEDIA_STANDARD: ISO 9660
ORIGINATOR: DEFENSE MAPPING AGENCY attn: ATISS
ADDRESSEE: HEADQUARTERS, DEFENSE MAPPING AGENCY ATTN: ATISS 8613 LEE HIGHWAY, FAIRFAX, VA 22031-2137
MEDIA_VOLUMES: 1
SEQ_NUMBERS: 1
NUM_DATA_SETS: 1
SECURITY_CLASS: U
DOWNGRADING: NO
DOWNGRADE_DATE: 00/00/0000 00:00:00
RELEASABILITY: DoD
OTHER_STD_NAME: NA
OTHER_STD_DATE: 00/00/0000 00:00:00
OTHER_STD_VER: NA
TRANSMITTAL_ID: 1
EDITION_NUMBER: 1
EDITION_DATE: 12/00/1992 00:00:00

TABLE 78. Library Attribute Table.

Table Definition:
Name: <ID> <Row Identifier> Type: I Count: 1 KeyType: N
Name: <LIBRARY_NAME> <Library name> Type: T Count: 8 KeyType: P
Name: <XMIN> <Westernmost longitude> Type: F Count: 1 KeyType: N
Name: <YMIN> <Southernmost latitude> Type: F Count: 1 KeyType: N
Name: <XMAX> <Easternmost longitude> Type: F Count: 1 KeyType: N
Name: <YMAX> <Northernmost latitude> Type: F Count: 1 KeyType: N

ID	LIBRARY_NAME	XMIN	YMIN	XMAX	YMAX
1	AXX08280	-76.449997	36.367001	-75.483002	37.400002
2	HXX08280	-76.453003	36.849998	-75.903999	37.141998
3	COASTAL	-76.050003	34.500000	-72.449997	41.583000
4	GENERAL	-76.682999	34.583000	-70.000000	44.159000
5	HXX07640	-74.286003	40.394001	-73.635002	40.896999
6	BROWSE	-180.000000	-90.000000	180.000000	90.000000

TABLE 79. Coverage Attribute Table (GENERAL Library).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: N
Name: <COVERAGE_NAME> <Coverage name> Type: T Count: 8 KeyType: P
Name: <DESCRIPTION> <Coverage description> Type: T Count: 50 KeyType: N
Name: <LEVEL> <Topology level> Type: I Count: 1 KeyType: N

ID	COVERAGE_NAME	DESCRIPTION	LEVEL
1	CUL	Cultural Landmarks	3
2	ECR	Earth Cover	3
3	HYD	Hydrography	3
4	IWY	Inland Waterways	3
5	LIM	Limits	3
6	NAV	Aids to Navigation	2
7	OBS	Obstructions	3
8	POR	Port Facilities	2
9	DQY	Data Quality	3

TABLE 80. Geographic Reference Table (GENERAL library).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <DATA_TYPE> <Data Type>  Type: T Count: 3 KeyType: N
Name: <UNITS> <Units>  Type: T Count: 3 KeyType: N
Name: <ELLIPSOID_NAME> <Ellipsoid>  Type: T Count: 15 KeyType: N
Name: <ELLIPSOID_DETAIL> <Ellipsoid Details>  Type: T Count: 50 KeyType: N
Name: <VERT_DATUM_NAME> <Datum Vertical Reference>  Type: T Count: 15 KeyType: N
Name: <VERT_DATUM_CODE> <Vertical Datum Category>  Type: T Count: 3 KeyType: N
Name: <SOUND_DATUM_NAME> <Sounding Datum>  Type: T Count: 15 KeyType: N
Name: <SOUND_DATUM_CODE> <Vertical Datum Category>  Type: T Count: 3 KeyType: N
Name: <GEO_DATUM_NAME> <Datum Geodetic Name>  Type: T Count: 15 KeyType: N
Name: <GEO_DATUM_CODE> <Datum Geodetic Code>  Type: T Count: 3 KeyType: N
Name: <PROJECTION_NAME> <Projection Name>  Type: T Count: 20 KeyType: N


ID: 1
DATA_TYPE: GEO
UNITS: DEG
ELLIPSOID_NAME: WGS84
ELLIPSOID_DETAIL: A=6378137,B=6356752M
VERT_DATUM_NAME: MEAN SEA LEVEL
VERT_DATUM_CODE: 015
SOUND_DATUM_NAME: MEAN LOWER LOW
SOUND_DATUM_CODE: 014
GEO_DATUM_NAME: WGS84
GEO_DATUM_CODE: WGE
PROJECTION_NAME: DECIMAL DEGREES
```

TABLE 81. Library Header Table (GENERAL Library).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <PRODUCT_TYPE> <Product Type>  Type: T Count: 12 KeyType: N
Name: <LIBRARY_NAME> <Name>  Type: T Count: 12 KeyType: N
Name: <DESCRIPTION> <Description of the library>  Type: T Count: 100 KeyType: N
Name: <DATA_STRUCT_CODE> <Data Structure Code>  Type: T Count: 1 KeyType: N
Name: <SCALE> <Denominator of representative fraction>  Type: I Count: 1 KeyType: N
Name: <SOURCE_SERIES> <Series>  Type: T Count: 15 KeyType: N
Name: <SOURCE_ID> <ID of the source reference>  Type: T Count: 30 KeyType: N
Name: <SOURCE_EDITION> <Edition number of the source>  Type: T Count: 20 KeyType: N
Name: <SOURCE_NAME> <Name of library source>  Type: T Count: 100 KeyType: N
Name: <SOURCE_DATE> <Source Date>  Type: D Count: 1 KeyType: N
Name: <SECURITY_CLASS> <Security Classification>  Type: T Count: 1 KeyType: N
Name: <DOWNGRADING> <Downgrading>  Type: T Count: 3 KeyType: N
Name: <DOWNGRADING_DATE> <Date>  Type: D Count: 1 KeyType: N
Name: <RELEASABILITY> <Releasability>  Type: T Count: 20 KeyType: N

ID: 1
PRODUCT_TYPE: DNC
LIBRARY_NAME: GENERAL
DESCRIPTION: This library contains digital nautical chart data extracted from NOS chart 13003
DATA_STRUCT_CODE: 8
SCALE: 1200000
SOURCE_SERIES: HAC
SOURCE_ID: NOS 13003
SOURCE_EDITION: 39
SOURCE_NAME: Cape Sable to Cape Hatteras
SOURCE_DATE: 05/25/1991 00:00:00
SECURITY_CLASS: U
DOWNGRADING: NO
DOWNGRADING_DATE: 00/00/0000 00:00:00
RELEASABILITY: DoD
```

TABLE 82. Line Feature Table (LIBREF coverage).

Table Definition:
Name: <ID> <Row Identifier> Type: I Count: 1 KeyType: P
Name: <EDG_ID> <Edge primitive ID> Type: I Count: 1 KeyType: N

ID	EDG_ID
1	73
2	248
3	253
4	170
5	24
6	63
7	345
8	100
9	7
10	9
11	25
12	112
13	245
14	62
15	31
16	37
17	53
18	152
19	177
20	220
21	247
22	312
23	301
24	15
25	72
26	95
27	164
28	57
29	205
30	199
31	266
32	35
33	19
34	24
35	303
36	47
37	128
38	126
39	49
40	207
41	333
42	228
43	235
44	37
45	115
46	77
47	93
48	412
49	385
50	176
51	190
52	235
53	47
54	53
55	96
56	103
57	119
58	134
59	220
60	260
61	52
62	14
63	20
64	345
65	531
66	487
67	200
68	209
69	114
70	53
71	67
72	153
73	49
74	228
75	137
76	277
77	19
78	344
79	171
80	173
81	62
82	172
83	92
84	98
85	83
86	44
87	126
88	152
89	175
90	122
91	229
92	165
93	73
94	96
95	195
96	184
97	72
98	283
99	353
100	428
101	127
102	411
103	96
104	79
105	52
106	104
107	25
108	109
109	27
110	33
111	54
112	221
113	244
114	153
115	111
116	27
117	73
118	206
119	37
120	127
121	39
122	233
123	155
124	32
125	52
126	66
127	62
128	23
129	162
130	199
131	165
132	43
133	172

TABLE 83. Area Feature Table (TILEREF coverage).

Table Definition:
Name: <ID> <Row Identifier> Type: I Count: 1 KeyType: P
Name: <TILE_NAME> <Tile Name> Type: T Count: 12 KeyType: N
Name: <FAC_ID> <Face primitive ID> Type: T Count: 1 KeyType: N

ID	TILE_NAME	FAC_ID
1	GKNA	2
2	HKAA	3
3	HKDA	4
4	GJNN	5
5	HJAN	6
6	HJDN	7
7	GJNK	8
8	HJAK	9
9	HJDK	10
10	GJNG	11
11	HJAG	12
12	HJDG	13
13	GJND	14
14	HJAD	15
15	HJDD	26

TABLE 84. Coverage Attribute Table (BROWSE library).

Table Definition:
Name: <ID> <Row Identifier> Type: I Count: 1 KeyType: N
Name: <COVERAGE_NAME> <Coverage name> Type: T Count: 8 KeyType: P
Name: <DESCRIPTION> <Coverage description> Type: T Count: 50 KeyType: N
Name: <LEVEL> <Topology level> Type: I Count: 1 KeyType: N

ID	COVERAGE_NAME	DESCRIPTION	LEVEL
1	COA	Coastlines/countries	3

TABLE 85. Geographic Reference Table (BROWSE library).

```
Table Definition:
Name: <ID> <Row Identifier>  Type: I Count: 1 KeyType: P
Name: <DATA_TYPE> <Data Type>  Type: T Count: 3 KeyType: N
Name: <UNITS> <Units>  Type: T Count: 3 KeyType: N
Name: <ELLIPSOID> <Ellipsoid>  Type: T Count: 15 KeyType: N
Name: <ELLIPSOID_DETAIL> <Ellipsoid Details>  Type: T Count: 50 KeyType: N
Name: <VERT_DATUM_REF> <Datum Vertical Reference>  Type: T Count: 15 KeyType:
N
Name: <VERT_DATUM_CODE> <Vertical Datum Code>  Type: T Count: 3 KeyType: N
Name: <SOUND_DATUM> <Sounding Datum>  Type: T Count: 15 KeyType: N
Name: <SOUND_DATUM_CODE> <Vertical Datum Code>  Type: T Count: 3 KeyType: N
Name: <GEO_DATUM_NAME> <Datum Geodetic Name>  Type: T Count: 15 KeyType: N
Name: <GEO_DATUM_CODE> <Datum Geodetic Code>  Type: T Count: 3 KeyType: N
Name: <PROJECTION_NAME> <Projection Name>  Type: T Count: 20 KeyType: N


ID: 1
DATA_TYPE: GEO
UNITS: DE
ELLIPSOID: WGS 84
ELLIPSOID_DETAIL: WGE 6378137,B=6356752 Meters
VERT_DATUM_REF: MEAN SEA LEVEL
VERT_DATUM_CODE: 015
SOUND_DATUM: MEAN SEA LEVEL
SOUND_DATUM_CODE: 015
GEO_DATUM_NAME: WGS 84
GEO_DATUM_CODE: WGE
PROJECTION_NAME:
```

TABLE 86. Library Header Table (BROWSE library).

```
Table Definition:
Name: <ID> <Row Identifier>  Type: I Count: 1 KeyType: P
Name: <PRODUCT_TYPE> <Product Type>  Type: T Count: 12 KeyType: N
Name: <LIBRARY_NAME> <Name>  Type: T Count: 12 KeyType: N
Name: <DESCRIPTION> <Description of the library>  Type: T Count: 100 KeyType: N
Name: <DATA_STRUCT_CODE> <Data Structure Code>  Type: T Count: 1 KeyType: N
Name: <SCALE> <Denominator of the representative fraction>  Type: I Count: 1      KeyType: N
Name: <SOURCE_SERIES> <Series>  Type: T Count: 15 KeyType: N
Name: <SOURCE_ID> <Identifier of the source reference>  Type: T Count: 30 KeyType: N
Name: <SOURCE_EDITION> <Edition number of the source>  Type: T Count: 20 KeyType: N
Name: <SOURCE_NAME> <Name of library source>  Type: T Count: 100 KeyType: N
Name: <SOURCE_DATE> <Source Date>  Type: D Count: 1 KeyType: N
Name: <SECURITY_CLASS> <Security Classification>  Type: T Count: 1 KeyType: N
Name: <DOWNGRADING> <Downgrading>  Type: T Count: 3 KeyType: N
Name: <DOWNGRADING_DATE> <Date>  Type: D Count: 1 KeyType: N
Name: <RELEASABILITY> <Releasability>  Type: T Count: 20 KeyType: N

ID: 1
PRODUCT_TYPE: DNC
LIBRARY_NAME: BROWSE
DESCRIPTION: The BROWSE library contains data which supports overview displays at a global
scale
DATA_STRUCT_CODE: 8
SCALE: 31000000
SOURCE_SERIES: ORG COMPILATION
SOURCE_ID: N/A
SOURCE_EDITION: ONE
SOURCE_NAME: ORIGINAL COMPILATION
SOURCE_DATE: 00/00/1992 00:00:00
SECURITY_CLASS: U
DOWNGRADING: NO
DOWNGRADING_DATE: 00/00/0000 00:00:00
RELEASABILITY: DoD
```

TABLE 87. Feature Class Schema Table (LIM coverage).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <FEATURE_CLASS> <Name of Feature Class>  Type: T Count: 8 KeyType: N
Name: <TABLE1> <First Table>  Type: T Count: 12 KeyType: N
Name: <TABLE1_KEY> <Column Name in First Table>  Type: T Count: 24 KeyType: N
Name: <TABLE2> <Second Table>  Type: T Count: 12 KeyType: N
Name: <TABLE2_KEY> <Column Name in Second Table>  Type: T Count: 24 KeyType: N

ID: 1
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.AFT
TABLE1_KEY: ID
TABLE2: LIMBNDYA.AJT
```

```
TABLE2_KEY: LIMBNDYA.AFT_ID

ID: 2
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.AJT
TABLE1_KEY: FAC_ID
TABLE2: FAC
TABLE2_KEY: ID

ID: 3
FEATURE_CLASS: LIMBNDYA
TABLE1: FAC
TABLE1_KEY: ID
TABLE2: LIMBNDYA.AJT
TABLE2_KEY: FAC_ID

ID: 4
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.AJT
TABLE1_KEY: LIMBNDYA.AFT_ID
TABLE2: LIMBNDYA.AFT
TABLE2_KEY: ID

ID: 5
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.AFT
TABLE1_KEY: ID
TABLE2: LIMBNDYA.NJT
TABLE2_KEY: FEATURE_ID

ID: 6
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.NJT
TABLE1_KEY: RAT_ID
TABLE2: NOTES.RAT
TABLE2_KEY: ID

ID: 7
FEATURE_CLASS: LIMBNDYA
TABLE1: NOTES.RAT
TABLE1_KEY: ID
TABLE2: LIMBNDYA.NJT
TABLE2_KEY: RAT_ID

ID: 8
FEATURE_CLASS: LIMBNDYA
TABLE1: LIMBNDYA.NJT
TABLE1_KEY: FEATURE_ID
TABLE2: LIMBNDYA.AFT
TABLE2_KEY: ID
```

TABLE 87. Feature Class Schema Table (LIM coverage)- Continued.

```
ID: 9
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.LFT
TABLE1_KEY: ID
TABLE2: LIMBNDYL.LJT
TABLE2_KEY: LIMBNDYL.LFT_ID
ID: 10
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.LJT
TABLE1_KEY: EDG_ID
TABLE2: EDG
TABLE2_KEY: ID

ID: 11
FEATURE_CLASS: LIMBNDYL
TABLE1: EDG
TABLE1_KEY: ID
TABLE2: LIMBNDYL.LJT
TABLE2_KEY: EDG_ID

ID: 12
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.LJT
TABLE1_KEY: LIMBNDYL.LFT_ID
TABLE2: LIMBNDYL.LFT
TABLE2_KEY: ID

ID: 13
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.LFT
TABLE1_KEY: ID
TABLE2: LIMBNDYL.NJT
TABLE2_KEY: FEATURE_ID

ID: 14
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.NJT
TABLE1_KEY: RAT_ID
TABLE2: NOTES.RAT
TABLE2_KEY: ID

ID: 15
FEATURE_CLASS: LIMBNDYL
TABLE1: NOTES.RAT
TABLE1_KEY: ID
TABLE2: LIMBNDYL.NJT
TABLE2_KEY: RAT_ID
```

TABLE 87. Feature Class Schema Table (LIM coverage)- Continued.

```
ID: 16
FEATURE_CLASS: LIMBNDYL
TABLE1: LIMBNDYL.NJT
TABLE1_KEY: FEATURE_ID
TABLE2: LIMBNDYL.LFT
TABLE2_KEY: ID
```

TABLE 87. Feature Class Schema Table (LIM coverage)- Continued.

TABLE 88. Area Feature Table (LIMBNDYA.AFT).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <F_CODE> <FACC Code>  Type: T Count: 5 KeyType: N VDT: <CHAR.VDT> TDX:
      <LIMBNDA1.ATI>
Name: <COD> <Certainty of Delineation>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>
Name: <HOC> <Hydrographic Origin Category>  Type: S Count: 1 KeyType: N VDT:  <INT.VDT>
Name: <MAC> <Maritime Area Category>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>
Name: <NAM> <Name>  Type: T Count: 70 KeyType: N
Name: <OPS> <Operational Status>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>
Name: <PBV> <Pilot Boarding Vehicle>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>
Name: <PRO> <Product Category>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>
Name: <USE> <Usage>  Type: S Count: 1 KeyType: N VDT: <INT.VDT>

ID: 1
F_CODE: FC036
COD: -32767
HOC: -32767
MAC: 43
NAM: N/A
OPS: -32767
PBV: -32767
PRO: 0
USE: 999

ID: 2
F_CODE: FC036
COD: -32767
HOC: -32767
MAC: 43
NAM: N/A
OPS: -32767
PBV: -32767
PRO: 0
USE: 999
```

```
ID: 3
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: DUMPING AREA CAUTION
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 4
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 5
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 6
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 2
PBV: 0
PRO: 33
USE: -32767
```

TABLE 88. Area Feature Table (LIMBNDYA.AFT) - Continued.

```
ID: 7
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 8
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 9
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: EXPLOSIVES DUMPING AREA
OPS: 1
PBV: 0
PRO: 33
USE: -32767

ID: 10
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 1
PBV: 0
PRO: 130
USE: -32767
```

TABLE 88. Area Feature Table (LIMBNDYA.AFT) - Continued.

```
ID: 11
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 130
USE: -32767

ID: 12
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 1
PBV: 0
PRO: 130
USE: -32767

ID: 13
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 130
USE: -32767

ID: 14
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 1
PBV: 0
PRO: 130
USE: -32767
```

TABLE 88. Area Feature Table (LIMBNDYA.AFT) - Continued.

```
ID: 15
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 13
USE: -32767

ID: 16
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 130
USE: -32767

ID: 17
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 1
PBV: 0
PRO: 130
USE: -32767

ID: 18
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 1
PBV: 0
PRO: 130
USE: -32767
```

TABLE 88. Area Feature Table (LIMBNDYA.AFT) - Continued.

```
ID: 19
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 130
USE: -32767

ID: 20
F_CODE: FC021
COD: 1
HOC: 4
MAC: 30
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 13
USE: -32767

ID: 21
F_CODE: FC021
COD: 1
HOC: 4
MAC: 999
NAM: DUMP SITE
OPS: 2
PBV: 0
PRO: 130
USE: -32767
```

TABLE 88. Area Feature Table (LIMBNDYA.AFT) - Continued.

TABLE 89. Area Join Table (LIMBNDYA.AJT).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <TILE_ID> <Tile Reference ID> Type: S Count: 1 KeyType: N TDX: <LIMBNDA1.JTI>
Name: <FAC_ID> <Primitive ID> Type: I Count: 1 KeyType: N TDX: <LIMBNDA2.JTI>
Name: <LIMBNDYA.AFT_ID> <Feature Table ID> Type: I Count: 1 KeyType: N TDX: <LIMBNDA3.JTI>

ID	TILE_ID	FAC_ID	LIMBNDYA.AFT_ID
1	7	2	10
2	7	3	11
3	9	2	13
4	9	3	14
5	9	4	15
6	9	5	16
7	9	6	17
8	10	2	12
9	10	3	1
10	10	4	2
11	10	5	3
12	10	6	4
13	12	2	18
14	12	3	5
15	12	4	19
16	12	5	6
17	12	6	20
18	12	7	21
19	12	8	7
20	12	9	8
21	12	10	9
22	13	2	18
23	13	3	6

TABLE 90. Notes Join Table (LIMBNDYA.NJT).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <FEATURE_ID> <Feature ID>  Type: I Count: 1 KeyType: N
Name: <RAT_ID> <Related Attribute ID>  Type: I Count: 1 KeyType: N


ID    FEATURE_ID RAT_ID
----- ---------- ------

    1          1      1
    2          2      1
    3         15      2
    4         18      2
    5         19      2
    6         20      2
    7         21      2
```

TABLE 91. Notes Related Attribute Table (LIM coverage).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <TEXT> <Feature Notes>  Type: T Count: * KeyType: N


ID: 1
TEXT: NOTE C  AREA TO BE AVOIDED All vessels carrying cargoes of oil or
hazardous materials and all other vessels of more than 1,000 gross tons should
avoid the area (MSC IMO XLIII/18).

ID: 2
TEXT: NOTE S  Regulations for Ocean Dumping Sites are contained in 40 CFR,
Parts 220-229.  Additional information concerning the regulations and
requirements for use of the sites may be obtained from Environmental
Protection Agency (EPA).  See U.S. Coast Pilots appendix for addresses of EPA
offices.
```

TABLE 92. Character Value Description Table (LIM coverage).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <TABLE> <Feature Class Table Name> Type: T Count: 12 KeyType: N
Name: <ATTRIBUTE> <Attribute Name> Type: T Count: 10 KeyType: N
Name: <VALUE> <Attribute Value> Type: T Count: 5 KeyType: N
Name: <DESCRIPTION> <Attribute Value Description> Type: T Count: 50 KeyType: N

```
ID: 1
TABLE: LIMBNDYA.AFT
ATTRIBUTE: F_CODE
VALUE: FC021
DESCRIPTION: Maritime Limit Boundary

ID: 2
TABLE: LIMBNDYA.AFT
ATTRIBUTE: F_CODE
VALUE: FC036
DESCRIPTION: Restricted Area

ID: 3
TABLE: MARITIMA.AFT
ATTRIBUTE: F_CODE
VALUE: FC031
DESCRIPTION: Maritime Area

ID: 4
TABLE: ROUTEA.AFT
ATTRIBUTE: F_CODE
VALUE: FC165
DESCRIPTION: Route (Maritime)

ID: 5
TABLE: SEPARTNA.AFT
ATTRIBUTE: F_CODE
VALUE: FC041
DESCRIPTION: Traffic Separation Scheme (TSS)

ID: 6
TABLE: LIMBNDYL.LFT
ATTRIBUTE: F_CODE
VALUE: FC021
DESCRIPTION: Maritime Limit Boundary

ID: 7
TABLE: LIMBNDYL.LFT
ATTRIBUTE: F_CODE
VALUE: FC036
DESCRIPTION: Restricted Area
```

ID: 8
TABLE: MARITIML.LFT
ATTRIBUTE: F_CODE
VALUE: FC031
DESCRIPTION: Maritime Area

ID: 9
TABLE: ROUTEL.LFT
ATTRIBUTE: F_CODE
VALUE: FC165
DESCRIPTION: Route (Maritime)

ID: 10
TABLE: SEPARTNL.LFT
ATTRIBUTE: F_CODE
VALUE: FC041
DESCRIPTION: Traffic Separation Scheme (TSS)

ID: 11
TABLE: LIMBNDYP.PFT
ATTRIBUTE: F_CODE
VALUE: FC021
DESCRIPTION: Maritime Limit Boundary

ID: 12
TABLE: LIMBNDYP.PFT
ATTRIBUTE: F_CODE
VALUE: FC036
DESCRIPTION: Restricted Area

ID: 13
TABLE: MARITIMP.PFT
ATTRIBUTE: F_CODE
VALUE: FC031
DESCRIPTION: Maritime Area

ID: 14
TABLE: ROUTEP.PFT
ATTRIBUTE: F_CODE
VALUE: FC165
DESCRIPTION: Route (Maritime)

ID: 15
TABLE: SEPARTNP.PFT
ATTRIBUTE: F_CODE
VALUE: FC041
DESCRIPTION: Traffic Separation Scheme (TSS)

TABLE 92. Character Value Description Table (LIM coverage)-Continued.

TABLE 93. Integer Value Description Table (LIM coverage).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <TABLE> <Feature Class Table Name> Type: T Count: 12 KeyType: N
Name: <ATTRIBUTE> <Attribute Name> Type: T Count: 10 KeyType: N
Name: <VALUE> <Attribute Value> Type: S Count: 1 KeyType: N
Name: <DESCRIPTION> <Attribute Value Description> Type: T Count: 50 KeyType: N

```
ID: 1
TABLE: LIMBNDYA.AFT
ATTRIBUTE: COD
VALUE: 1
DESCRIPTION: Limits and Info Known

ID: 2
TABLE: LIMBNDYA.AFT
ATTRIBUTE: COD
VALUE: 2
DESCRIPTION: Limits and Info Unknown

ID: 3
TABLE: LIMBNDYA.AFT
ATTRIBUTE: HOC
VALUE: 4
DESCRIPTION: Man-made

ID: 4
TABLE: LIMBNDYA.AFT
ATTRIBUTE: HOC
VALUE: 5
DESCRIPTION: Natural

ID: 5
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 15
DESCRIPTION: Anchoring Prohibited

ID: 6
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 20
DESCRIPTION: Submarine Cable Area

ID: 7
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 21
DESCRIPTION: Pipeline Area
```

ID: 8
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 22
DESCRIPTION: Fishing Prohibited

ID: 9
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 23
DESCRIPTION: Cable and Pipeline Area

ID: 10
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 26
DESCRIPTION: Unsurveyed Area

ID: 11
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 30
DESCRIPTION: Dumping Ground for Hazardous Materials

ID: 12
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 31
DESCRIPTION: Incineraton Area

ID: 13
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 32
DESCRIPTION: Oil Field

ID: 14
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 33
DESCRIPTION: Gas Field

ID: 15
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 37
DESCRIPTION: Safety Zone

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 16
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 43
DESCRIPTION: Areas to be avoided

ID: 17
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 48
DESCRIPTION: Pilot Boarding Area

ID: 18
TABLE: LIMBNDYA.AFT
ATTRIBUTE: MAC
VALUE: 999
DESCRIPTION: Other

ID: 19
TABLE: LIMBNDYA.AFT
ATTRIBUTE: OPS
VALUE: 1
DESCRIPTION: Operational

ID: 20
TABLE: LIMBNDYA.AFT
ATTRIBUTE: OPS
VALUE: 2
DESCRIPTION: Non-Operational

ID: 21
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PBV
VALUE: 1
DESCRIPTION: By boat

ID: 22
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PBV
VALUE: 2
DESCRIPTION: By helicopter

ID: 23
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 0
DESCRIPTION: Unknown

TABLE 93. <u>Integer Value Description Table (LIM coverage)</u> - Continued.

ID: 24
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 3
DESCRIPTION: Ammunition

ID: 25
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 13
DESCRIPTION: Chemical

ID: 26
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 33
DESCRIPTION: Explosives

ID: 27
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 38
DESCRIPTION: Gas

ID: 28
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 39
DESCRIPTION: Gasoline

ID: 29
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 67
DESCRIPTION: Oil

ID: 30
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 82
DESCRIPTION: Radioactive Material

ID: 31
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 116
DESCRIPTION: Water

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 32
TABLE: LIMBNDYA.AFT
ATTRIBUTE: PRO
VALUE: 130
DESCRIPTION: None

ID: 33
TABLE: LIMBNDYA.AFT
ATTRIBUTE: USE
VALUE: 51
DESCRIPTION: Telegraph

ID: 34
TABLE: LIMBNDYA.AFT
ATTRIBUTE: USE
VALUE: 52
DESCRIPTION: Telephone

ID: 35
TABLE: LIMBNDYA.AFT
ATTRIBUTE: USE
VALUE: 53
DESCRIPTION: Power

ID: 36
TABLE: LIMBNDYA.AFT
ATTRIBUTE: USE
VALUE: 999
DESCRIPTION: Other

ID: 37
TABLE: MARITIMA.AFT
ATTRIBUTE: ATN
VALUE: 1
DESCRIPTION: Marked

ID: 38
TABLE: MARITIMA.AFT
ATTRIBUTE: ATN
VALUE: 2
DESCRIPTION: Unmarked

ID: 39
TABLE: MARITIMA.AFT
ATTRIBUTE: COD
VALUE: 1
DESCRIPTION: Limits and Info Known

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 40
TABLE: MARITIMA.AFT
ATTRIBUTE: COD
VALUE: 2
DESCRIPTION: Limits and Info Unknown

ID: 41
TABLE: MARITIMA.AFT
ATTRIBUTE: DAT
VALUE: 26
DESCRIPTION: Information as of _____

ID: 42
TABLE: MARITIMA.AFT
ATTRIBUTE: EXS
VALUE: 1
DESCRIPTION: Definite

ID: 43
TABLE: MARITIMA.AFT
ATTRIBUTE: EXS
VALUE: 3
DESCRIPTION: Reported

ID: 44
TABLE: MARITIMA.AFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 45
TABLE: MARITIMA.AFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

ID: 46
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 2
DESCRIPTION: Dredged Channel / Dredged Area

ID: 47
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 4
DESCRIPTION: Mine Danger Area

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 48
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 5
DESCRIPTION: Prohibited Shipping Area / Entry Prohibited

ID: 49
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 9
DESCRIPTION: Work in Progress Area

ID: 50
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 40
DESCRIPTION: Roundabout Zone (TSS)

ID: 51
TABLE: MARITIMA.AFT
ATTRIBUTE: MAC
VALUE: 41
DESCRIPTION: Inshore Traffic Zone (TSS)

ID: 52
TABLE: MARITIMA.AFT
ATTRIBUTE: MAS
VALUE: 1
DESCRIPTION: Maintained

ID: 53
TABLE: MARITIMA.AFT
ATTRIBUTE: MAS
VALUE: 2
DESCRIPTION: Not Maintained

ID: 54
TABLE: MARITIMA.AFT
ATTRIBUTE: TSP
VALUE: 3
DESCRIPTION: Separation Zone Area

ID: 55
TABLE: MARITIMA.AFT
ATTRIBUTE: WPC
VALUE: 1
DESCRIPTION: Land Reclamation

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 56
TABLE: ROUTEA.AFT
ATTRIBUTE: HDI
VALUE: 9
DESCRIPTION: Depth Known by Other Than Wire

ID: 57
TABLE: ROUTEA.AFT
ATTRIBUTE: HDI
VALUE: 12
DESCRIPTION: Depth Unknown

ID: 58
TABLE: ROUTEA.AFT
ATTRIBUTE: RTT
VALUE: 11
DESCRIPTION: Two-way Route

ID: 59
TABLE: SEPARTNA.AFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 60
TABLE: SEPARTNA.AFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

ID: 61
TABLE: SEPARTNA.AFT
ATTRIBUTE: TSP
VALUE: 3
DESCRIPTION: Separation Zone Area

ID: 62
TABLE: SEPARTNA.AFT
ATTRIBUTE: TSP
VALUE: 6
DESCRIPTION: Inbound Area

ID: 63
TABLE: SEPARTNA.AFT
ATTRIBUTE: TSP
VALUE: 7
DESCRIPTION: Outbound Area

TABLE 93. <u>Integer Value Description Table (LIM coverage)</u> - Continued.

ID: 64
TABLE: LIMBNDYL.LFT
ATTRIBUTE: COD
VALUE: 1
DESCRIPTION: Limits and Info Known

ID: 65
TABLE: LIMBNDYL.LFT
ATTRIBUTE: COD
VALUE: 2
DESCRIPTION: Limits and Info Unknown

ID: 66
TABLE: LIMBNDYL.LFT
ATTRIBUTE: HOC
VALUE: 4
DESCRIPTION: Man-made

ID: 67
TABLE: LIMBNDYL.LFT
ATTRIBUTE: HOC
VALUE: 5
DESCRIPTION: Natural

ID: 68
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 20
DESCRIPTION: Submarine Cable Area

ID: 69
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 21
DESCRIPTION: Pipeline Area

ID: 70
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 23
DESCRIPTION: Cable and Pipeline Area

ID: 71
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 26
DESCRIPTION: Unsurveyed Area

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 72
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 30
DESCRIPTION: Dumping Ground for Hazardous Materials

ID: 73
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 31
DESCRIPTION: Incineration Area

ID: 74
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 32
DESCRIPTION: Oil Field

ID: 75
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 33
DESCRIPTION: Gas Field

ID: 76
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 48
DESCRIPTION: Pilot Boarding Area

ID: 77
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MAC
VALUE: 999
DESCRIPTION: Other

ID: 78
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MBL
VALUE: 6
DESCRIPTION: Territorial Waters-Limit of Sovereignty

ID: 79
TABLE: LIMBNDYL.LFT
ATTRIBUTE: MBL
VALUE: 7
DESCRIPTION: Territorial Waters Baseline

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 80
TABLE: LIMBNDYL.LFT
ATTRIBUTE: OPS
VALUE: 1
DESCRIPTION: Operational

ID: 81
TABLE: LIMBNDYL.LFT
ATTRIBUTE: OPS
VALUE: 2
DESCRIPTION: Non-Operational

ID: 82
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PBV
VALUE: 1
DESCRIPTION: By boat

ID: 83
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PBV
VALUE: 2
DESCRIPTION: By helicopter

ID: 84
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 0
DESCRIPTION: Unknown

ID: 85
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 3
DESCRIPTION: Ammunition

ID: 86
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 13
DESCRIPTION: Chemical

ID: 87
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 33
DESCRIPTION: Explosives

TABLE 93. <u>Integer Value Description Table (LIM coverage)</u> - Continued.

ID: 88
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 38
DESCRIPTION: Gas

ID: 89
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 39
DESCRIPTION: Gasoline

ID: 90
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 67
DESCRIPTION: Oil

ID: 91
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 82
DESCRIPTION: Radioactive Material

ID: 92
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 116
DESCRIPTION: Water

ID: 93
TABLE: LIMBNDYL.LFT
ATTRIBUTE: PRO
VALUE: 130
DESCRIPTION: None

ID: 94
TABLE: LIMBNDYL.LFT
ATTRIBUTE: USE
VALUE: 51
DESCRIPTION: Telegraph

ID: 95
TABLE: LIMBNDYL.LFT
ATTRIBUTE: USE
VALUE: 52
DESCRIPTION: Telephone

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 96
TABLE: LIMBNDYL.LFT
ATTRIBUTE: USE
VALUE: 53
DESCRIPTION: Power

ID: 97
TABLE: LIMBNDYL.LFT
ATTRIBUTE: USE
VALUE: 999
DESCRIPTION: Other

ID: 98
TABLE: MARITIML.LFT
ATTRIBUTE: ATN
VALUE: 1
DESCRIPTION: Marked

ID: 99
TABLE: MARITIML.LFT
ATTRIBUTE: ATN
VALUE: 2
DESCRIPTION: Unmarked

ID: 100
TABLE: MARITIML.LFT
ATTRIBUTE: COD
VALUE: 1
DESCRIPTION: Limits and Info Known

ID: 101
TABLE: MARITIML.LFT
ATTRIBUTE: COD
VALUE: 2
DESCRIPTION: Limits and Info Unknown

ID: 102
TABLE: MARITIML.LFT
ATTRIBUTE: DAT
VALUE: 26
DESCRIPTION: Information as of _____

ID: 103
TABLE: MARITIML.LFT
ATTRIBUTE: EXS
VALUE: 1
DESCRIPTION: Definite

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 104
TABLE: MARITIML.LFT
ATTRIBUTE: EXS
VALUE: 3
DESCRIPTION: Reported

ID: 105
TABLE: MARITIML.LFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 106
TABLE: MARITIML.LFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

ID: 107
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 2
DESCRIPTION: Dredged Channel / Dredged Area

ID: 108
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 4
DESCRIPTION: Mine Danger Area

ID: 109
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 5
DESCRIPTION: Prohibited Shipping Area / Entry Prohibited

ID: 110
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 9
DESCRIPTION: Work in Progress Area

ID: 111
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 40
DESCRIPTION: Roundabout Zone (TSS)

TABLE 93. <u>Integer Value Description Table (LIM coverage)</u> - Continued.

ID: 112
TABLE: MARITIML.LFT
ATTRIBUTE: MAC
VALUE: 41
DESCRIPTION: Inshore Traffic Zone (TSS)

ID: 113
TABLE: MARITIML.LFT
ATTRIBUTE: MAS
VALUE: 1
DESCRIPTION: Maintained

ID: 114
TABLE: MARITIML.LFT
ATTRIBUTE: MAS
VALUE: 2
DESCRIPTION: Not Maintained

ID: 115
TABLE: MARITIML.LFT
ATTRIBUTE: TSP
VALUE: 2
DESCRIPTION: Outer Boundary

ID: 116
TABLE: MARITIML.LFT
ATTRIBUTE: WPC
VALUE: 2
DESCRIPTION: Construction of Structures

ID: 117
TABLE: ROUTEL.LFT
ATTRIBUTE: ATN
VALUE: 1
DESCRIPTION: Marked

ID: 118
TABLE: ROUTEL.LFT
ATTRIBUTE: ATN
VALUE: 2
DESCRIPTION: Unmarked

ID: 119
TABLE: ROUTEL.LFT
ATTRIBUTE: EXS
VALUE: 22
DESCRIPTION: One-Way

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 120
TABLE: ROUTEL.LFT
ATTRIBUTE: EXS
VALUE: 23
DESCRIPTION: Two-way

ID: 121
TABLE: ROUTEL.LFT
ATTRIBUTE: HDI
VALUE: 9
DESCRIPTION: Depth Known by Other Than Wire

ID: 122
TABLE: ROUTEL.LFT
ATTRIBUTE: HDI
VALUE: 12
DESCRIPTION: Depth Unknown

ID: 123
TABLE: ROUTEL.LFT
ATTRIBUTE: RTT
VALUE: 2
DESCRIPTION: Recommended track for other than deepdraft vessels

ID: 124
TABLE: ROUTEL.LFT
ATTRIBUTE: RTT
VALUE: 3
DESCRIPTION: Recommended track for deep draft vessels

ID: 125
TABLE: ROUTEL.LFT
ATTRIBUTE: RTT
VALUE: 5
DESCRIPTION: Transit Route

ID: 126
TABLE: SEPARTNL.LFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 127
TABLE: SEPARTNL.LFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 128
TABLE: SEPARTNL.LFT
ATTRIBUTE: TSP
VALUE: 2
DESCRIPTION: Outer Boundary

ID: 129
TABLE: SEPARTNL.LFT
ATTRIBUTE: TSP
VALUE: 4
DESCRIPTION: Separation Zone Line

ID: 130
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 15
DESCRIPTION: Anchoring Prohibited

ID: 131
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 22
DESCRIPTION: Fishing Prohibited

ID: 132
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 37
DESCRIPTION: Safety Zone

ID: 133
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 43
DESCRIPTION: Area to be Avoided

ID: 134
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 48
DESCRIPTION: Pilot Boarding Area

ID: 135
TABLE: LIMBNDYP.PFT
ATTRIBUTE: MAC
VALUE: 999
DESCRIPTION: Other

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 136
TABLE: LIMBNDYP.PFT
ATTRIBUTE: PBV
VALUE: 1
DESCRIPTION: By boat

ID: 137
TABLE: LIMBNDYP.PFT
ATTRIBUTE: PBV
VALUE: 2
DESCRIPTION: By helicopter

ID: 138
TABLE: MARITIMP.PFT
ATTRIBUTE: EXS
VALUE: 1
DESCRIPTION: Definite

ID: 139
TABLE: MARITIMP.PFT
ATTRIBUTE: EXS
VALUE: 3
DESCRIPTION: Reported

ID: 140
TABLE: MARITIMP.PFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 141
TABLE: MARITIMP.PFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

ID: 142
TABLE: MARITIMP.PFT
ATTRIBUTE: MAC
VALUE: 2
DESCRIPTION: Dredged Channel / Dredged Area

ID: 143
TABLE: MARITIMP.PFT
ATTRIBUTE: MAC
VALUE: 4
DESCRIPTION: Mine Danger Area

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 144
TABLE: MARITIMP.PFT
ATTRIBUTE: MAC
VALUE: 5
DESCRIPTION: Prohibited Shipping Area / Entry Prohibited

ID: 145
TABLE: MARITIMP.PFT
ATTRIBUTE: MAC
VALUE: 40
DESCRIPTION: Roundabout Zone (TSS)

ID: 146
TABLE: MARITIMP.PFT
ATTRIBUTE: MAS
VALUE: 1
DESCRIPTION: Maintained

ID: 147
TABLE: MARITIMP.PFT
ATTRIBUTE: TSP
VALUE: 1
DESCRIPTION: Arrow

ID: 148
TABLE: MARITIMP.PFT
ATTRIBUTE: TSP
VALUE: 5
DESCRIPTION: Separation Zone Point

ID: 149
TABLE: ROUTEP.PFT
ATTRIBUTE: RTT
VALUE: 13
DESCRIPTION: Recommended Direction of Traffic Flow

ID: 150
TABLE: SEPARTNP.PFT
ATTRIBUTE: IAS
VALUE: 1
DESCRIPTION: Approved

ID: 151
TABLE: SEPARTNP.PFT
ATTRIBUTE: IAS
VALUE: 2
DESCRIPTION: Not Approved

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

ID: 152
TABLE: SEPARTNP.PFT
ATTRIBUTE: TSP
VALUE: 1
DESCRIPTION: Arrow

TABLE 93. Integer Value Description Table (LIM coverage) - Continued.

TABLE 94. Feature Class Attribute Table (LIM coverage).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <FCLASS> <Feature Class Name> Type: T Count: 8 KeyType: U
Name: <TYPE> <Feature Class Type> Type: T Count: 1 KeyType: N
Name: <DESCR> <Feature Class Description> Type: T Count: * KeyType: N

ID: 1
FCLASS: LIMBNDYA
TYPE: A
DESCR: Limit Boundary Areas

ID: 2
FCLASS: MARITIMA
TYPE: A
DESCR: Maritime Areas

ID: 3
FCLASS: ROUTEA
TYPE: A
DESCR: Route Areas

ID: 4
FCLASS: SEPARTNA
TYPE: A
DESCR: Separation Areas

ID: 5
FCLASS: LIMBNDYL
TYPE: L
DESCR: Limit Boundary Lines

ID: 6
FCLASS: MARITIML
TYPE: L
DESCR: Maritime Lines

ID: 7
FCLASS: ROUTEL
TYPE: L
DESCR: Route LinesID: 8
FCLASS: SEPARTNL
TYPE: L
DESCR: Separation Lines

ID: 9
FCLASS: LIMBNDYP
TYPE: P
DESCR: Limit Boundary Points

ID: 10
FCLASS: MARITIMP
TYPE: P
DESCR: Maritime Points

ID: 11
FCLASS: ROUTEP
TYPE: P
DESCR: Route Points

ID: 12
FCLASS: SEPARTNP
TYPE: P
DESCR: Separation Points

TABLE 94. Feature Class Attribute Table (LIM coverage) - Continued.

TABLE 95. Face Feature Index Table (LIM coverage).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <PRIM_ID> <Primitive ID> Type: I Count: 1 KeyType: N
Name: <TILE_ID> <Tile Reference ID> Type: S Count: 1 KeyType: N TDX: <FACFIT1.FTI>
Name: <FC_ID> <Feature Class ID> Type: I Count: 1 KeyType: N TDX: <FACFIT2.FTI>
Name: <FEATURE_ID> <Feature ID> Type: I Count: 1 KeyType: N

ID	PRIM_ID	TILE_ID	FC_ID	FEATURE_ID
1	2	7	1	10
2	3	7	1	11
3	2	9	1	13
4	3	9	1	14
5	4	9	1	15
6	5	9	1	16
7	6	9	1	17
8	2	10	1	12
9	3	10	1	1
10	4	10	1	2
11	5	10	1	3
12	6	10	1	4
13	2	12	1	18
14	3	12	1	5
15	4	12	1	19
16	5	12	1	6
17	6	12	1	20
18	7	12	1	21
19	8	12	1	7
20	9	12	1	8
21	10	12	1	9
22	2	13	1	18
23	3	13	1	6

TABLE 96. Entity Node Table (ECR coverage, tile GJND).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <CONTAINING_FACE> <Foreign Key to Face Table> Type: I Count: 1 KeyType: N
Name: <FIRST_EDGE> <Foreign Key to Edge Table (null)> Type: X Count: 1 KeyType: N
Name: <COORDINATE> <Coordinate of Node> Type: C Count: 1 KeyType: N

ID	CONTAINING_FACE	FIRST_EDGE	COORDINATE
1	1		(-75.621017, 35.913723)
2	1		(-75.613991, 35.901527)
3	1		(-75.624588, 35.897240)
4	1		(-75.614906, 35.886475)
5	1		(-75.578033, 35.816120)
6	1		(-75.566589, 35.806774)
7	1		(-75.493271, 35.668743)
8	1		(-75.487999, 35.659386)
9	1		(-75.482750, 35.639961)
10	1		(-76.656609, 35.381500)
11	1		(-76.047752, 35.377773)
12	1		(-76.035400, 35.374870)
13	1		(-76.067184, 35.368416)
14	1		(-76.587822, 35.366974)
15	1		(-76.395523, 35.340000)
16	1		(-76.548210, 35.328655)
17	1		(-75.808258, 35.177292)
18	1		(-76.412537, 35.077507)
19	1		(-76.015900, 35.073334)
20	1		(-76.028259, 35.071178)
21	1		(-76.406387, 35.065910)
22	1		(-76.275963, 34.999775)
23	1		(-76.302475, 34.985313)
24	1		(-76.431297, 34.819256)
25	1		(-76.610695, 34.717709)
26	1		(-76.620407, 34.713360)
27	1		(-76.603661, 34.706787)
28	1		(-76.611610, 34.703884)
29	1		(-76.542885, 34.663757)
30	1		(-76.533180, 34.662289)
31	1		(-76.544762, 34.620064)

TABLE 97. Connected Node Table (ECR coverage, tile GJND).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <CONTAINING_FACE> <Null column> Type: X Count: 1 KeyType: N
Name: <FIRST_EDGE> <Foriegn Key to Edge table> Type: I Count: 1 KeyType: N
Name: <COORDINATE> <Node x and y location> Type: C Count: 1 KeyType: N

ID	CONTAINING_FACE	FIRST_EDGE	COORDINATE
1		1	(-76.682999, 36.000000)
2		1	(-75.724709, 36.000000)
3		2	(-75.695251, 36.000000)
4		3	(-75.688568, 36.000000)
5		4	(-75.658348, 36.000000)
6		5	(-75.000000, 36.000000)
7		8	(-75.939995, 35.970375)
8		7	(-76.682999, 35.938274)
9		9	(-75.642799, 35.837349)
10		12	(-75.562187, 35.801792)
11		10	(-75.586731, 35.799744)
12		13	(-76.682999, 35.502422)
13		14	(-76.682999, 35.492249)
14		15	(-76.682999, 35.415573)
15		16	(-76.068169, 35.405346)
16		18	(-76.410057, 35.346630)
17		19	(-76.682999, 35.341675)
18		20	(-76.364342, 35.335648)
19		22	(-76.275993, 35.313805)
20		23	(-75.699867, 35.218925)
21		24	(-76.682999, 35.162468)
22		26	(-76.530022, 35.148819)
23		27	(-76.682999, 35.144127)
24		28	(-75.999313, 35.078228)
25		29	(-76.044540, 35.066784)
26		34	(-76.080490, 35.066425)
27		30	(-76.428391, 35.065540)
28		31	(-76.036873, 35.062855)
29		35	(-76.067741, 35.045845)
30		32	(-76.682999, 35.028908)
31		37	(-76.129501, 35.002449)
32		36	(-76.153053, 34.992039)
33		38	(-76.258011, 34.977470)
34		39	(-76.263695, 34.969795)
35		40	(-76.682999, 34.950218)
36		41	(-76.207153, 34.942383)
37		42	(-76.319107, 34.850117)
38		50	(-76.332199, 34.845810)

39	43	(-76.682999, 34.767570)
40	44	(-76.489479, 34.751442)
41	46	(-76.608307, 34.682129)
42	47	(-76.535240, 34.678097)
43	48	(-76.590057, 34.675552)
44	49	(-76.536140, 34.637466)
45	51	(-75.000000, 34.583000)

TABLE 97. Connected Node Table (ECR coverage, tile GJND) - Conntinued.

TABLE 98. Edge Table (ECR coverage, tile GJND).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <START_NODE> <Start/Left Node>  Type: I Count: 1 KeyType: N
Name: <END_NODE> <End/Right Node>  Type: I Count: 1 KeyType: N
Name: <RIGHT_FACE> <Right Face>  Type: K Count: 1 KeyType: N
Name: <LEFT_FACE> <Left Face>  Type: K Count: 1 KeyType: N
Name: <RIGHT_EDGE> <Right Edge from End Node>  Type: K Count: 1 KeyType: N
Name: <LEFT_EDGE> <Left Edge from Start Node>  Type: K Count: 1 KeyType: N
Name: <COORDINATES> <Coordinates of Edge>  Type: C Count: * KeyType: N

ID: 1
START_NODE: 1
END_NODE: 2
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 6, tile: 0, exid: 0)
LEFT_EDGE: (id: 7, tile: 0, exid: 0)
COORDINATES: (-76.682999,36.000000)
             (-75.724709,36.000000)

ID: 2
START_NODE: 2
END_NODE: 3
RIGHT_FACE: (id: 3, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 6, tile: 0, exid: 0)
LEFT_EDGE: (id: 1, tile: 0, exid: 0)
COORDINATES: (-75.724709,36.000000)
             (-75.695251,36.000000)
```

```
ID: 3
START_NODE: 3
END_NODE: 4
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 11, tile: 0, exid: 0)
LEFT_EDGE: (id: 2, tile: 0, exid: 0)
COORDINATES: (-75.695251,36.000000)
             (-75.688568,36.000000)

ID: 4
START_NODE: 4
END_NODE: 5
RIGHT_FACE: (id: 4, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 11, tile: 0, exid: 0)
LEFT_EDGE: (id: 3, tile: 11, exid: 111)
COORDINATES: (-75.688568,36.000000)
             (-75.658348,36.000000)

ID: 5
START_NODE: 5
END_NODE: 6
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 51, tile: 0, exid: 0)
LEFT_EDGE: (id: 4, tile: 0, exid: 0)
COORDINATES: (-75.658348,36.000000)
             (-75.000000,36.000000)

ID: 6
START_NODE: 2
END_NODE: 3
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 3, tile: 0, exid: 0)
RIGHT_EDGE: (id: 3, tile: 0, exid: 0)
LEFT_EDGE: (id: 2, tile: 0, exid: 0)
COORDINATES: (-75.724709,36.000000)
             (-75.720421,35.998920)
             (-75.708244,35.997810)
             (-75.695366,35.998447)
             (-75.695251,36.000000)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
ID: 7
START_NODE: 8
END_NODE: 1
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 1, tile: 0, exid: 0)
LEFT_EDGE: (id: 13, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.938274)
             (-76.682999,36.000000)

ID: 8
START_NODE: 7
END_NODE: 7
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 6, tile: 0, exid: 0)
RIGHT_EDGE: (id: 8, tile: 0, exid: 0)
LEFT_EDGE: (id: 8, tile: 0, exid: 0)
COORDINATES: (-75.939995,35.970375)
             (-75.944977,35.965706)
             (-75.944931,35.958733)
             (-75.932671,35.944256)
             (-75.923332,35.938480)
             (-75.920464,35.937912)
             (-75.914749,35.940262)
             (-75.900513,35.952526)
             (-75.874092,35.963680)
             (-75.869843,35.970669)
             (-75.869881,35.977058)
             (-75.874199,35.981689)
             (-75.884941,35.978550)
             (-75.905510,35.968105)
             (-75.905693,35.980392)
             (-75.912689,35.977261)
             (-75.937126,35.970387)
             (-75.939995,35.970375)

ID: 9
START_NODE: 9
END_NODE: 9
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 7, tile: 0, exid: 0)
RIGHT_EDGE: (id: 9, tile: 0, exid: 0)
LEFT_EDGE: (id: 9, tile: 0, exid: 0)
COORDINATES: (-75.642799,35.837349)
             (-75.635651,35.837963)
             (-75.629250,35.845554)
             (-75.631470,35.857182)
             (-75.637970,35.866467)
             (-75.640160,35.873440)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-75.639450,35.874603)
               (-75.624344,35.861866)
               (-75.620049,35.862469)
               (-75.618668,35.871201)
               (-75.636726,35.897301)
               (-75.641075,35.906006)
               (-75.642517,35.907745)
               (-75.642548,35.911816)
               (-75.646141,35.914707)
               (-75.650429,35.913525)
               (-75.665421,35.907066)
               (-75.669014,35.908794)
               (-75.665535,35.925671)
               (-75.682037,35.931995)
               (-75.697807,35.935417)
               (-75.706429,35.939453)
               (-75.724350,35.943443)
               (-75.729347,35.941681)
               (-75.734314,35.934685)
               (-75.730690,35.927727)
               (-75.709114,35.910954)
               (-75.705460,35.899338)
               (-75.683838,35.874416)
               (-75.649979,35.840229)
               (-75.642799,35.837349)

ID: 10
START_NODE: 11
END_NODE: 11
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 8, tile: 0, exid: 0)
RIGHT_EDGE: (id: 10, tile: 0, exid: 0)
LEFT_EDGE: (id: 10, tile: 0, exid: 0)
COORDINATES: (-75.586731,35.799744)
               (-75.583893,35.803833)
               (-75.586777,35.807312)
               (-75.590363,35.807880)
               (-75.595352,35.804367)
               (-75.592461,35.799721)
               (-75.586731,35.799744)

ID: 11
START_NODE: 5
END_NODE: 4
RIGHT_FACE: (id: 4, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 4, tile: 0, exid: 0)
LEFT_EDGE: (id: 5, tile: 0, exid: 0)
COORDINATES: (-75.658348,36.000000)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-75.639351,35.974602)
               (-75.617714,35.947960)
               (-75.605423,35.928829)
               (-75.592400,35.906208)
               (-75.585144,35.890537)
               (-75.569221,35.862099)
               (-75.560524,35.845261)
               (-75.548248,35.827267)
               (-75.536285,35.802868)
               (-75.535164,35.795300)
               (-75.538742,35.794704)
               (-75.555267,35.804531)
               (-75.572556,35.821926)
               (-75.589157,35.843975)
               (-75.602203,35.870106)
               (-75.605957,35.898594)
               (-75.613922,35.914261)
               (-75.616814,35.918316)
               (-75.619713,35.924702)
               (-75.639893,35.946705)
               (-75.668686,35.972153)
               (-75.683083,35.985458)
               (-75.685280,35.993580)
               (-75.688568,36.000000)

ID: 12
START_NODE: 10
END_NODE: 10
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 9, tile: 0, exid: 0)
RIGHT_EDGE: (id: 12, tile: 0, exid: 0)
LEFT_EDGE: (id: 12, tile: 0, exid: 0)
COORDINATES: (-75.562187,35.801792)
             (-75.560860,35.796886)
             (-75.554840,35.793606)
             (-75.553482,35.799603)
             (-75.558167,35.802879)
             (-75.562187,35.801792)

ID: 13
START_NODE: 12
END_NODE: 8
RIGHT_FACE: (id: 5, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 21, tile: 0, exid: 0)
LEFT_EDGE: (id: 14, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.502422)
             (-76.682999,35.938274)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
ID: 14
START_NODE: 13
END_NODE: 12
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 21, tile: 0, exid: 0)
LEFT_EDGE: (id: 15, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.492249)
             (-76.682999,35.502422)

ID: 15
START_NODE: 14
END_NODE: 13
RIGHT_FACE: (id: 11, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 17, tile: 0, exid: 0)
LEFT_EDGE: (id: 19, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.415573)
             (-76.682999,35.492249)

ID: 16
START_NODE: 15
END_NODE: 15
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 12, tile: 0, exid: 0)
RIGHT_EDGE: (id: 16, tile: 0, exid: 0)
LEFT_EDGE: (id: 16, tile: 0, exid: 0)
COORDINATES: (-76.068169,35.405346)
             (-76.072853,35.405354)
             (-76.076889,35.401524)
             (-76.077568,35.397141)
             (-76.068855,35.397129)
             (-76.065498,35.400959)
             (-76.068169,35.405346)

ID: 17
START_NODE: 14
END_NODE: 13
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 11, tile: 0, exid: 0)
RIGHT_EDGE: (id: 14, tile: 0, exid: 0)
LEFT_EDGE: (id: 15, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.415573)
             (-76.670822,35.411346)
             (-76.665825,35.414295)
             (-76.659416,35.419586)
             (-76.661835,35.429855)
             (-76.642517,35.431107)
             (-76.626129,35.426941)
             (-76.642906,35.423721)
```

TABLE 98. <u>Edge Table (ECR coverage, tile GJND)</u> - Continued.

```
             (-76.650978,35.417900)
             (-76.654526,35.413204)
             (-76.653061,35.407356)
             (-76.629387,35.399265)
             (-76.594269,35.391804)
             (-76.579208,35.387768)
             (-76.566322,35.387825)
             (-76.569962,35.397175)
             (-76.583244,35.410530)
             (-76.592659,35.428043)
             (-76.594193,35.444416)
             (-76.592133,35.459045)
             (-76.593575,35.460796)
             (-76.591209,35.476643)
             (-76.599129,35.484207)
             (-76.610603,35.487667)
             (-76.617775,35.490559)
             (-76.623543,35.496380)
             (-76.626457,35.504551)
             (-76.633652,35.510365)
             (-76.647247,35.509724)
             (-76.682999,35.492249)

ID: 18
START_NODE: 16
END_NODE: 16
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 13, tile: 0, exid: 0)
RIGHT_EDGE: (id: 18, tile: 0, exid: 0)
LEFT_EDGE: (id: 18, tile: 0, exid: 0)
COORDINATES: (-76.410057,35.346630)
             (-76.407204,35.347225)
             (-76.405052,35.347237)
             (-76.403633,35.349586)
             (-76.406517,35.352501)
             (-76.410080,35.349556)
             (-76.410057,35.346630)

ID: 19
START_NODE: 17
END_NODE: 14
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 17, tile: 0, exid: 0)
LEFT_EDGE: (id: 24, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.341675)
             (-76.682999,35.415573)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
ID: 20
START_NODE: 18
END_NODE: 18
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 15, tile: 0, exid: 0)
RIGHT_EDGE: (id: 20, tile: 0, exid: 0)
LEFT_EDGE: (id: 20, tile: 0, exid: 0)
COORDINATES: (-76.364342,35.335648)
             (-76.356384,35.342133)
             (-76.347641,35.343422)
             (-76.343666,35.346016)
             (-76.347427,35.348064)
             (-76.361725,35.344784)
             (-76.366707,35.343445)
             (-76.368309,35.339550)
             (-76.364342,35.335648)

ID: 21
START_NODE: 8
END_NODE: 12
RIGHT_FACE: (id: 5, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 13, tile: 0, exid: 0)
LEFT_EDGE: (id: 7, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.938274)
             (-76.680702,35.938282)
             (-76.672821,35.937153)
             (-76.627007,35.937347)
             (-76.546051,35.949108)
             (-76.516014,35.954464)
             (-76.450256,35.970428)
             (-76.431114,35.977680)
             (-76.413971,35.984726)
             (-76.403954,35.985348)
             (-76.401794,35.983032)
             (-76.394600,35.977833)
             (-76.385918,35.962181)
             (-76.384354,35.941269)
             (-76.395035,35.931339)
             (-76.394302,35.929600)
             (-76.381592,35.928154)
             (-76.370705,35.933765)
             (-76.351418,35.939659)
             (-76.338524,35.938549)
             (-76.322739,35.932808)
             (-76.316986,35.928761)
             (-76.296204,35.924194)
             (-76.290482,35.925964)
             (-76.281868,35.921932)
             (-76.280502,35.920109)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

(-76.273056,35.918949)
(-76.266708,35.917652)
(-76.263535,35.915066)
(-76.258942,35.905788)
(-76.256393,35.913120)
(-76.257973,35.916348)
(-76.262718,35.923450)
(-76.278603,35.924118)
(-76.287331,35.928646)
(-76.296844,35.935108)
(-76.320702,35.942856)
(-76.319290,35.946346)
(-76.312080,35.953541)
(-76.295761,35.964298)
(-76.267691,35.977203)
(-76.235031,35.987797)
(-76.215721,35.991364)
(-76.173508,35.995605)
(-76.140572,35.993420)
(-76.124107,35.993488)
(-76.108345,35.991230)
(-76.081856,35.991924)
(-76.055344,35.986805)
(-76.038841,35.981064)
(-76.025162,35.968918)
(-76.021523,35.958477)
(-76.022903,35.950336)
(-76.029320,35.946819)
(-76.042213,35.947929)
(-76.057327,35.960068)
(-76.064438,35.953068)
(-76.065819,35.944340)
(-76.060745,35.934483)
(-76.064262,35.920841)
(-76.051422,35.931618)
(-76.046425,35.933380)
(-76.039230,35.923851)
(-76.027107,35.935783)
(-76.019951,35.935818)
(-76.014191,35.930027)
(-76.016281,35.920715)
(-76.020515,35.911976)
(-76.028328,35.900894)
(-76.036110,35.886906)
(-76.039619,35.875256)
(-76.056755,35.868202)
(-76.070358,35.864399)
(-76.083260,35.867256)
(-76.088264,35.865490)
(-76.087532,35.862579)

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.068146,35.853352)
(-76.063812,35.850716)
(-76.063004,35.836170)
(-76.066551,35.830338)
(-76.066498,35.822769)
(-76.060730,35.815224)
(-76.057083,35.804760)
(-76.053307,35.772743)
(-76.054718,35.769241)
(-76.059723,35.768059)
(-76.064034,35.770954)
(-76.080490,35.769135)
(-76.089104,35.772594)
(-76.090523,35.771423)
(-76.086914,35.763027)
(-76.069740,35.763683)
(-76.056808,35.756165)
(-76.053223,35.758762)
(-76.045288,35.749477)
(-76.042381,35.741329)
(-76.042343,35.736084)
(-76.049522,35.738968)
(-76.055916,35.731365)
(-76.058014,35.723198)
(-76.060310,35.703644)
(-76.054344,35.708641)
(-76.045769,35.711006)
(-76.040733,35.706367)
(-76.038475,35.688297)
(-76.041298,35.681873)
(-76.063499,35.683529)
(-76.077812,35.683472)
(-76.092064,35.684433)
(-76.102165,35.685116)
(-76.113632,35.686821)
(-76.121544,35.693783)
(-76.123383,35.700771)
(-76.130898,35.700741)
(-76.139496,35.702454)
(-76.152412,35.707062)
(-76.171791,35.700291)
(-76.167732,35.697243)
(-76.154388,35.702190)
(-76.147354,35.700089)
(-76.140884,35.692837)
(-76.126717,35.693707)
(-76.122925,35.684444)
(-76.105965,35.675667)
(-76.102814,35.674034)
(-76.087067,35.674099)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.078461,35.671803)
(-76.072739,35.672409)
(-76.057701,35.672474)
(-76.039009,35.658554)
(-76.026176,35.667355)
(-76.016998,35.688972)
(-76.004295,35.717594)
(-76.000862,35.741505)
(-75.993118,35.763092)
(-75.990326,35.775341)
(-75.990601,35.820179)
(-75.986420,35.838818)
(-75.990799,35.851604)
(-75.994469,35.866138)
(-75.985901,35.869663)
(-75.989586,35.887100)
(-75.987450,35.888851)
(-75.983902,35.894104)
(-75.969772,35.896393)
(-75.955872,35.905689)
(-75.943993,35.923927)
(-75.924049,35.923176)
(-75.918289,35.917969)
(-75.922951,35.909512)
(-75.903908,35.906979)
(-75.897423,35.900612)
(-75.897362,35.890724)
(-75.894493,35.889572)
(-75.885216,35.894264)
(-75.875183,35.892559)
(-75.865799,35.879803)
(-75.863647,35.879814)
(-75.861519,35.882149)
(-75.866592,35.892597)
(-75.884521,35.898338)
(-75.912788,35.927261)
(-75.912804,35.930172)
(-75.906837,35.935127)
(-75.889656,35.935200)
(-75.871796,35.924557)
(-75.861069,35.926346)
(-75.856697,35.930103)
(-75.848106,35.930141)
(-75.845947,35.927826)
(-75.834465,35.923805)
(-75.819466,35.928516)
(-75.841705,35.936562)
(-75.858192,35.940563)
(-75.872513,35.940502)
(-75.876823,35.943390)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-75.874695,35.946888)
(-75.866127,35.949825)
(-75.862564,35.952168)
(-75.856911,35.964977)
(-75.859856,35.977745)
(-75.859879,35.981812)
(-75.819695,35.966293)
(-75.795181,35.937920)
(-75.781502,35.924606)
(-75.773499,35.903706)
(-75.767731,35.897331)
(-75.755501,35.886913)
(-75.752541,35.871223)
(-75.741631,35.843338)
(-75.731552,35.834068)
(-75.732933,35.826496)
(-75.723534,35.811398)
(-75.721306,35.798599)
(-75.726250,35.787514)
(-75.730453,35.772934)
(-75.728981,35.765369)
(-75.729591,35.747887)
(-75.720909,35.733356)
(-75.718018,35.729286)
(-75.714363,35.716480)
(-75.711357,35.693752)
(-75.717041,35.686150)
(-75.728462,35.680851)
(-75.740631,35.680801)
(-75.741402,35.689545)
(-75.747894,35.697685)
(-75.762238,35.702286)
(-75.774376,35.696987)
(-75.780746,35.685299)
(-75.777840,35.678314)
(-75.769249,35.678352)
(-75.763504,35.674873)
(-75.762741,35.667877)
(-75.760559,35.661472)
(-75.756218,35.654488)
(-75.744705,35.645203)
(-75.734573,35.626572)
(-75.746445,35.617653)
(-75.745224,35.613106)
(-75.752037,35.605358)
(-75.765900,35.599594)
(-75.770912,35.599571)
(-75.781937,35.595455)
(-75.782295,35.587849)
(-75.780853,35.585518)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

(-75.780800,35.577927)
(-75.783646,35.574413)
(-75.800819,35.573170)
(-75.809387,35.569630)
(-75.840866,35.567162)
(-75.840179,35.571255)
(-75.832184,35.574593)
(-75.813461,35.579788)
(-75.816650,35.587116)
(-75.818825,35.590614)
(-75.822411,35.592350)
(-75.842407,35.584675)
(-75.856720,35.583447)
(-75.862465,35.586342)
(-75.870430,35.602070)
(-75.874115,35.618401)
(-75.868401,35.620758)
(-75.864861,35.627777)
(-75.872032,35.628914)
(-75.883514,35.633533)
(-75.887077,35.631187)
(-75.885620,35.627689)
(-75.882690,35.616032)
(-75.886192,35.603172)
(-75.898705,35.596203)
(-75.895416,35.589706)
(-75.883919,35.583332)
(-75.884598,35.577492)
(-75.895279,35.568104)
(-75.913132,35.560436)
(-75.919540,35.555153)
(-75.920212,35.548141)
(-75.910904,35.548180)
(-75.913033,35.544666)
(-75.923027,35.539948)
(-75.942322,35.534611)
(-75.950890,35.530483)
(-75.952980,35.521126)
(-75.957977,35.518768)
(-75.969467,35.525146)
(-75.971603,35.523972)
(-75.972992,35.516953)
(-75.986603,35.518063)
(-75.989449,35.514545)
(-75.976532,35.509338)
(-75.963608,35.503548)
(-75.960709,35.497715)
(-75.964256,35.492439)
(-75.983551,35.486511)
(-75.997101,35.478271)

TABLE 98. <u>Edge Table (ECR coverage, tile GJND)</u> - Continued.

```
(-76.002045,35.467728)
(-75.998375,35.453121)
(-76.000496,35.449017)
(-76.013351,35.443115)
(-76.019768,35.438992)
(-76.014626,35.417366)
(-76.018196,35.416767)
(-76.025322,35.411472)
(-76.032478,35.410858)
(-76.036079,35.413769)
(-76.037544,35.420197)
(-76.034698,35.423134)
(-76.035431,35.425472)
(-76.046898,35.427765)
(-76.054794,35.430656)
(-76.061279,35.437649)
(-76.064140,35.437637)
(-76.069092,35.428253)
(-76.061920,35.425945)
(-76.059746,35.421276)
(-76.066879,35.417732)
(-76.071159,35.414791)
(-76.078270,35.407150)
(-76.081100,35.401875)
(-76.083908,35.392498)
(-76.078163,35.389011)
(-76.073868,35.389030)
(-76.067406,35.386127)
(-76.065956,35.383209)
(-76.068100,35.382614)
(-76.080185,35.369099)
(-76.092354,35.368462)
(-76.110199,35.360775)
(-76.128792,35.357769)
(-76.135971,35.360081)
(-76.140976,35.359474)
(-76.140938,35.353031)
(-76.136620,35.350124)
(-76.145172,35.343060)
(-76.146561,35.336025)
(-76.134010,35.332073)
(-76.147255,35.332508)
(-76.160133,35.331284)
(-76.165161,35.334190)
(-76.165947,35.345901)
(-76.176697,35.347610)
(-76.183151,35.349926)
(-76.191010,35.346966)
(-76.201744,35.346336)
(-76.211067,35.348637)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.221794,35.348007)
(-76.231110,35.349140)
(-76.236877,35.355556)
(-76.238350,35.362576)
(-76.241257,35.369591)
(-76.247719,35.373077)
(-76.253479,35.377735)
(-76.238235,35.393951)
(-76.256386,35.385334)
(-76.261375,35.381802)
(-76.259163,35.371273)
(-76.253357,35.357830)
(-76.257614,35.351368)
(-76.269035,35.346054)
(-76.285530,35.351254)
(-76.299133,35.352367)
(-76.305595,35.355267)
(-76.306328,35.358192)
(-76.304207,35.361713)
(-76.294907,35.362926)
(-76.284142,35.358871)
(-76.282028,35.364151)
(-76.285637,35.368820)
(-76.291359,35.368793)
(-76.306442,35.376343)
(-76.313629,35.381580)
(-76.313683,35.389191)
(-76.316574,35.393860)
(-76.329468,35.395561)
(-76.335213,35.399048)
(-76.339539,35.403713)
(-76.346039,35.413048)
(-76.348892,35.411282)
(-76.344536,35.401352)
(-76.340919,35.395512)
(-76.344444,35.386131)
(-76.344376,35.375011)
(-76.341484,35.370338)
(-76.342186,35.367996)
(-76.338531,35.356300)
(-76.342056,35.353809)
(-76.351410,35.354488)
(-76.357864,35.356804)
(-76.375061,35.359657)
(-76.385788,35.357857)
(-76.388634,35.354332)
(-76.392921,35.353142)
(-76.400887,35.368332)
(-76.415230,35.370487)
(-76.407394,35.376961)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.395927,35.377132)
(-76.362930,35.365566)
(-76.355072,35.368233)
(-76.362328,35.384888)
(-76.370941,35.387779)
(-76.382393,35.387142)
(-76.388878,35.395309)
(-76.392242,35.407963)
(-76.389397,35.410900)
(-76.378586,35.409641)
(-76.380058,35.422894)
(-76.384392,35.429375)
(-76.388382,35.431007)
(-76.394867,35.437416)
(-76.403763,35.448704)
(-76.407722,35.451958)
(-76.415726,35.454288)
(-76.416420,35.450191)
(-76.413513,35.443184)
(-76.406998,35.430927)
(-76.414124,35.425632)
(-76.418427,35.426785)
(-76.421989,35.424431)
(-76.422668,35.417992)
(-76.437584,35.398617)
(-76.446846,35.392139)
(-76.452568,35.390945)
(-76.457603,35.395607)
(-76.449028,35.419506)
(-76.457657,35.403801)
(-76.462997,35.399387)
(-76.474861,35.407238)
(-76.480576,35.406044)
(-76.481270,35.402531)
(-76.479103,35.399612)
(-76.473305,35.387932)
(-76.467499,35.373905)
(-76.471764,35.369205)
(-76.478203,35.369179)
(-76.491135,35.376148)
(-76.501938,35.387810)
(-76.502693,35.389969)
(-76.501480,35.404362)
(-76.506180,35.398605)
(-76.509743,35.396252)
(-76.515465,35.395058)
(-76.526962,35.402618)
(-76.532738,35.410198)
(-76.532784,35.417805)
(-76.527100,35.425438)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.511360,35.426674)
(-76.514244,35.430172)
(-76.544350,35.435894)
(-76.558731,35.446945)
(-76.558754,35.451042)
(-76.555222,35.458076)
(-76.545944,35.463379)
(-76.528038,35.481503)
(-76.533043,35.480896)
(-76.539925,35.476368)
(-76.548828,35.466873)
(-76.555283,35.468014)
(-76.566780,35.475571)
(-76.570442,35.488415)
(-76.579796,35.497143)
(-76.586281,35.504131)
(-76.586304,35.507053)
(-76.584160,35.508232)
(-76.583458,35.510574)
(-76.577003,35.508846)
(-76.551231,35.508369)
(-76.532600,35.505527)
(-76.526161,35.506138)
(-76.508972,35.504456)
(-76.498955,35.504498)
(-76.495522,35.491478)
(-76.493652,35.502529)
(-76.485672,35.508205)
(-76.470726,35.511036)
(-76.471558,35.519051)
(-76.470650,35.523590)
(-76.470261,35.540676)
(-76.456680,35.552605)
(-76.452026,35.556698)
(-76.462791,35.561325)
(-76.483360,35.576248)
(-76.487640,35.573311)
(-76.484039,35.570408)
(-76.478294,35.566929)
(-76.477493,35.552330)
(-76.483177,35.546463)
(-76.516296,35.531307)
(-76.523445,35.530109)
(-76.527756,35.531841)
(-76.549255,35.536427)
(-76.560699,35.534626)
(-76.581314,35.538815)
(-76.622818,35.536884)
(-76.649872,35.550690)
(-76.653488,35.555931)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
              (-76.658516,35.558830)
              (-76.662079,35.556480)
              (-76.662758,35.550049)
              (-76.648781,35.539513)
              (-76.640869,35.533123)
              (-76.636513,35.523205)
              (-76.643150,35.521759)
              (-76.658859,35.514679)
              (-76.680359,35.502712)
              (-76.682999,35.502422)

ID: 22
START_NODE: 19
END_NODE: 19
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 16, tile: 0, exid: 0)
RIGHT_EDGE: (id: 22, tile: 0, exid: 0)
LEFT_EDGE: (id: 22, tile: 0, exid: 0)
COORDINATES: (-76.275993,35.313805)
              (-76.270271,35.315002)
              (-76.273918,35.326115)
              (-76.281143,35.336044)
              (-76.284721,35.336613)
              (-76.288292,35.334843)
              (-76.288223,35.323715)
              (-76.281006,35.314369)
              (-76.275993,35.313805)

ID: 23
START_NODE: 20
END_NODE: 20
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 10, tile: 0, exid: 0)
RIGHT_EDGE: (id: 23, tile: 0, exid: 0)
LEFT_EDGE: (id: 23, tile: 0, exid: 0)
COORDINATES: (-75.699867,35.218925)
              (-75.709816,35.207150)
              (-75.721237,35.202408)
              (-75.730530,35.200024)
              (-75.732666,35.197666)
              (-75.730133,35.192982)
              (-75.716919,35.197731)
              (-75.691925,35.207813)
              (-75.655479,35.219112)
              (-75.631912,35.227421)
              (-75.619041,35.229824)
              (-75.604042,35.235165)
              (-75.581856,35.237015)
              (-75.570412,35.237652)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-75.531021,35.234886)
(-75.516724,35.238464)
(-75.513176,35.243172)
(-75.513214,35.249622)
(-75.514671,35.253719)
(-75.513367,35.274830)
(-75.509354,35.321724)
(-75.502342,35.345768)
(-75.494598,35.366295)
(-75.485451,35.392677)
(-75.484467,35.465809)
(-75.480995,35.483952)
(-75.474701,35.507359)
(-75.474739,35.514374)
(-75.469147,35.535431)
(-75.468491,35.545948)
(-75.467102,35.552963)
(-75.467148,35.559975)
(-75.464432,35.583927)
(-75.463051,35.592106)
(-75.464554,35.603775)
(-75.463928,35.618370)
(-75.466125,35.626534)
(-75.476334,35.656830)
(-75.478554,35.669071)
(-75.485146,35.692955)
(-75.486710,35.714520)
(-75.491798,35.727905)
(-75.494720,35.737801)
(-75.508469,35.761047)
(-75.512802,35.766853)
(-75.518532,35.767998)
(-75.527107,35.765629)
(-75.516533,35.754513)
(-75.515762,35.746105)
(-75.514015,35.731308)
(-75.511055,35.725407)
(-75.513138,35.705082)
(-75.508751,35.690525)
(-75.499390,35.681812)
(-75.486443,35.671371)
(-75.477356,35.648838)
(-75.474129,35.614998)
(-75.476585,35.607220)
(-75.475494,35.584446)
(-75.473923,35.564537)
(-75.475883,35.550694)
(-75.487473,35.520222)
(-75.487061,35.485718)
(-75.494286,35.456532)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-75.495110,35.444851)
               (-75.492180,35.426365)
               (-75.505630,35.371490)
               (-75.520287,35.352718)
               (-75.520050,35.314064)
               (-75.517700,35.280670)
               (-75.519081,35.273045)
               (-75.524788,35.270092)
               (-75.532669,35.271233)
               (-75.534088,35.268295)
               (-75.562714,35.267586)
               (-75.573448,35.266953)
               (-75.592072,35.268635)
               (-75.599915,35.263912)
               (-75.602051,35.261559)
               (-75.611320,35.255657)
               (-75.629829,35.238575)
               (-75.645515,35.228539)
               (-75.668365,35.219055)
               (-75.681992,35.223103)
               (-75.684883,35.227783)
               (-75.692039,35.226585)
               (-75.699867,35.218925)

ID: 24
START_NODE: 21
END_NODE: 17
RIGHT_FACE: (id: 14, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 25, tile: 0, exid: 0)
LEFT_EDGE: (id: 27, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.162468)
             (-76.682999,35.341675)

ID: 25
START_NODE: 21
END_NODE: 17
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 14, tile: 0, exid: 0)
RIGHT_EDGE: (id: 19, tile: 0, exid: 0)
LEFT_EDGE: (id: 24, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.162468)
             (-76.680435,35.162270)
             (-76.676865,35.163460)
             (-76.671875,35.167004)
             (-76.667625,35.175240)
             (-76.657654,35.183498)
             (-76.654800,35.184097)
             (-76.640457,35.180637)
             (-76.628288,35.180687)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.616158,35.187191)
(-76.601913,35.198402)
(-76.594063,35.203129)
(-76.587631,35.204330)
(-76.586212,35.207268)
(-76.595566,35.214855)
(-76.599159,35.216599)
(-76.598457,35.218948)
(-76.593452,35.219555)
(-76.584831,35.214901)
(-76.574043,35.207321)
(-76.572601,35.205566)
(-76.551804,35.198612)
(-76.543930,35.198647)
(-76.533844,35.188126)
(-76.528107,35.186390)
(-76.525993,35.192268)
(-76.526741,35.197544)
(-76.517487,35.206387)
(-76.519653,35.209309)
(-76.523956,35.211048)
(-76.533989,35.212769)
(-76.546921,35.219753)
(-76.561256,35.223213)
(-76.569115,35.219658)
(-76.570557,35.221413)
(-76.566643,35.226414)
(-76.574928,35.234882)
(-76.579247,35.237797)
(-76.580688,35.240139)
(-76.574249,35.240749)
(-76.567810,35.240780)
(-76.547760,35.239689)
(-76.500389,35.219364)
(-76.492523,35.220570)
(-76.494713,35.227596)
(-76.499023,35.231098)
(-76.510506,35.235741)
(-76.521278,35.241562)
(-76.526329,35.247990)
(-76.524910,35.249752)
(-76.510551,35.242779)
(-76.506493,35.249775)
(-76.497704,35.251820)
(-76.490089,35.260395)
(-76.487984,35.266853)
(-76.479195,35.265778)
(-76.472023,35.264046)
(-76.468460,35.265820)
(-76.471344,35.269329)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

(-76.468536,35.279305)
(-76.472168,35.287495)
(-76.476486,35.291580)
(-76.481491,35.289799)
(-76.484322,35.285099)
(-76.486465,35.283916)
(-76.486435,35.279228)
(-76.492882,35.279785)
(-76.498634,35.284451)
(-76.501564,35.295574)
(-76.498779,35.307892)
(-76.494507,35.312008)
(-76.484489,35.312054)
(-76.480919,35.314411)
(-76.481651,35.316753)
(-76.498138,35.320198)
(-76.528923,35.320656)
(-76.531769,35.317711)
(-76.532448,35.311852)
(-76.525970,35.305431)
(-76.535271,35.304810)
(-76.539543,35.301273)
(-76.539482,35.291313)
(-76.541611,35.288376)
(-76.543129,35.301846)
(-76.553200,35.310005)
(-76.551788,35.314114)
(-76.546791,35.316479)
(-76.549683,35.319981)
(-76.553268,35.321724)
(-76.576195,35.324554)
(-76.576225,35.329826)
(-76.604881,35.332634)
(-76.612022,35.330261)
(-76.616272,35.323215)
(-76.616196,35.310326)
(-76.621887,35.304443)
(-76.619713,35.299763)
(-76.598007,35.298771)
(-76.599556,35.296967)
(-76.612267,35.295685)
(-76.618637,35.289841)
(-76.624603,35.280991)
(-76.643211,35.279739)
(-76.656090,35.278515)
(-76.658241,35.279091)
(-76.657532,35.280853)
(-76.633224,35.285057)
(-76.626099,35.290363)
(-76.627563,35.296215)

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
              (-76.629730,35.299725)
              (-76.643013,35.305931)
              (-76.639297,35.306919)
              (-76.631935,35.309086)
              (-76.632744,35.323730)
              (-76.641350,35.327209)
              (-76.653519,35.325989)
              (-76.652809,35.327747)
              (-76.631409,35.340427)
              (-76.632874,35.344814)
              (-76.650078,35.348843)
              (-76.659370,35.347046)
              (-76.666504,35.342918)
              (-76.671509,35.341724)
              (-76.682999,35.341675)

ID: 26
START_NODE: 22
END_NODE: 22
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 19, tile: 0, exid: 0)
RIGHT_EDGE: (id: 26, tile: 0, exid: 0)
LEFT_EDGE: (id: 26, tile: 0, exid: 0)
COORDINATES: (-76.530022,35.148819)
              (-76.527184,35.152355)
              (-76.531509,35.157619)
              (-76.535088,35.157604)
              (-76.537216,35.154072)
              (-76.532883,35.148808)
              (-76.530022,35.148819)

ID: 27
START_NODE: 23
END_NODE: 21
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 25, tile: 0, exid: 0)
LEFT_EDGE: (id: 33, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.144127)
              (-76.682999,35.162468)

ID: 28
START_NODE: 24
END_NODE: 24
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 17, tile: 0, exid: 0)
RIGHT_EDGE: (id: 28, tile: 0, exid: 0)
LEFT_EDGE: (id: 28, tile: 0, exid: 0)
COORDINATES: (-75.999313,35.078228)
              (-75.971886,35.100964)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-75.949059,35.113396)
               (-75.930496,35.121696)
               (-75.878342,35.138943)
               (-75.861183,35.143127)
               (-75.839767,35.152023)
               (-75.826920,35.158535)
               (-75.769783,35.179905)
               (-75.761215,35.184635)
               (-75.762672,35.188152)
               (-75.772690,35.186935)
               (-75.793404,35.179806)
               (-75.841972,35.161991)
               (-75.874130,35.152466)
               (-75.899124,35.142380)
               (-75.948433,35.129253)
               (-75.969147,35.121532)
               (-75.978462,35.122665)
               (-75.982033,35.121479)
               (-75.990593,35.116741)
               (-75.991272,35.110279)
               (-75.989067,35.100891)
               (-75.998299,35.088512)
               (-76.002548,35.080860)
               (-75.999313,35.078228)

ID: 29
START_NODE: 25
END_NODE: 25
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 23, tile: 0, exid: 0)
RIGHT_EDGE: (id: 29, tile: 0, exid: 0)
LEFT_EDGE: (id: 29, tile: 0, exid: 0)
COORDINATES: (-76.044540,35.066784)
             (-76.043846,35.070896)
             (-76.048866,35.072052)
             (-76.051003,35.070866)
             (-76.049553,35.067936)
             (-76.044540,35.066784)

ID: 30
START_NODE: 27
END_NODE: 27
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 21, tile: 0, exid: 0)
RIGHT_EDGE: (id: 30, tile: 0, exid: 0)
LEFT_EDGE: (id: 30, tile: 0, exid: 0)
COORDINATES: (-76.428391,35.065540)
             (-76.428429,35.071419)
             (-76.432747,35.075512)
             (-76.436325,35.075497)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
              (-76.438469,35.074314)
              (-76.436279,35.068447)
              (-76.430534,35.065533)
              (-76.428391,35.065540)

ID: 31
START_NODE: 28
END_NODE: 28
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 24, tile: 0, exid: 0)
RIGHT_EDGE: (id: 31, tile: 0, exid: 0)
LEFT_EDGE: (id: 31, tile: 0, exid: 0)
COORDINATES: (-76.036873,35.062855)
              (-76.038879,35.063412)
              (-76.040894,35.062313)
              (-76.038895,35.059555)
              (-76.044266,35.055164)
              (-76.053650,35.054626)
              (-76.059021,35.050232)
              (-76.058357,35.048031)
              (-76.052322,35.048023)
              (-76.040916,35.054058)
              (-76.033531,35.059551)
              (-76.036873,35.062855)

ID: 32
START_NODE: 30
END_NODE: 23
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 18, tile: 0, exid: 0)
RIGHT_EDGE: (id: 27, tile: 0, exid: 0)
LEFT_EDGE: (id: 33, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.028908)
              (-76.677475,35.028931)
              (-76.658928,35.040180)
              (-76.643272,35.054939)
              (-76.617546,35.062691)
              (-76.601845,35.070396)
              (-76.595436,35.076298)
              (-76.592644,35.086887)
              (-76.597694,35.093327)
              (-76.601440,35.093193)
              (-76.608597,35.092575)
              (-76.629959,35.085712)
              (-76.636879,35.088188)
              (-76.609871,35.095039)
              (-76.596283,35.096859)
              (-76.590561,35.098061)
              (-76.579109,35.098694)
              (-76.562683,35.104637)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-76.545647,35.128204)
               (-76.543571,35.140541)
               (-76.540024,35.145252)
               (-76.542198,35.149357)
               (-76.543655,35.154045)
               (-76.554443,35.162804)
               (-76.560944,35.170994)
               (-76.572884,35.168732)
               (-76.585312,35.176758)
               (-76.589592,35.173809)
               (-76.589523,35.163246)
               (-76.580879,35.153889)
               (-76.575859,35.152149)
               (-76.574448,35.138985)
               (-76.578018,35.137794)
               (-76.580132,35.148022)
               (-76.586586,35.150928)
               (-76.593758,35.152660)
               (-76.600952,35.154068)
               (-76.611702,35.160801)
               (-76.627449,35.159561)
               (-76.638580,35.154453)
               (-76.641792,35.160355)
               (-76.646805,35.159744)
               (-76.654694,35.167076)
               (-76.660416,35.165878)
               (-76.671585,35.154408)
               (-76.682999,35.144127)

ID: 33
START_NODE: 30
END_NODE: 23
RIGHT_FACE: (id: 18, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 32, tile: 0, exid: 0)
LEFT_EDGE: (id: 40, tile: 0, exid: 0)
COORDINATES: (-76.682999,35.028908)
             (-76.682999,35.144127)

ID: 34
START_NODE: 26
END_NODE: 26
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 22, tile: 0, exid: 0)
RIGHT_EDGE: (id: 34, tile: 0, exid: 0)
LEFT_EDGE: (id: 34, tile: 0, exid: 0)
COORDINATES: (-76.080490,35.066425)
             (-76.104668,35.040462)
             (-76.120331,35.025696)
             (-76.123154,35.019806)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-76.120995,35.018051)
               (-76.115288,35.020428)
               (-76.109612,35.029270)
               (-76.093224,35.041687)
               (-76.089676,35.047577)
               (-76.067574,35.061779)
               (-76.058327,35.071220)
               (-76.061928,35.074734)
               (-76.066940,35.074711)
               (-76.080490,35.066425)

ID: 35
START_NODE: 29
END_NODE: 29
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 25, tile: 0, exid: 0)
RIGHT_EDGE: (id: 35, tile: 0, exid: 0)
LEFT_EDGE: (id: 35, tile: 0, exid: 0)
COORDINATES: (-76.067741,35.045845)
               (-76.075790,35.043102)
               (-76.090569,35.029366)
               (-76.108025,35.016731)
               (-76.116081,35.010136)
               (-76.114075,35.007931)
               (-76.105347,35.015625)
               (-76.083862,35.029358)
               (-76.069107,35.035938)
               (-76.063728,35.042534)
               (-76.067741,35.045845)

ID: 36
START_NODE: 32
END_NODE: 32
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 26, tile: 0, exid: 0)
RIGHT_EDGE: (id: 36, tile: 0, exid: 0)
LEFT_EDGE: (id: 36, tile: 0, exid: 0)
COORDINATES: (-76.153053,34.992039)
               (-76.148071,34.996765)
               (-76.140251,35.006207)
               (-76.140976,35.007381)
               (-76.145271,35.007362)
               (-76.152390,35.000862)
               (-76.156639,34.992611)
               (-76.153053,34.992039)

ID: 37
START_NODE: 31
END_NODE: 31
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
LEFT_FACE: (id: 28, tile: 0, exid: 0)
RIGHT_EDGE: (id: 37, tile: 0, exid: 0)
LEFT_EDGE: (id: 37, tile: 0, exid: 0)
COORDINATES: (-76.129501,35.002449)
             (-76.140251,34.992554)
             (-76.155693,34.979908)
             (-76.151680,34.978249)
             (-76.146309,34.982098)
             (-76.141602,34.988148)
             (-76.134888,34.992542)
             (-76.128166,35.000793)
             (-76.129501,35.002449)

ID: 38
START_NODE: 33
END_NODE: 33
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 29, tile: 0, exid: 0)
RIGHT_EDGE: (id: 38, tile: 0, exid: 0)
LEFT_EDGE: (id: 38, tile: 0, exid: 0)
COORDINATES: (-76.258011,34.977470)
             (-76.251602,34.982792)
             (-76.251640,34.989262)
             (-76.258133,34.996883)
             (-76.262421,34.996861)
             (-76.265266,34.993912)
             (-76.266655,34.985672)
             (-76.263031,34.978626)
             (-76.261589,34.977451)
             (-76.258011,34.977470)

ID: 39
START_NODE: 34
END_NODE: 34
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 27, tile: 0, exid: 0)
RIGHT_EDGE: (id: 39, tile: 0, exid: 0)
LEFT_EDGE: (id: 39, tile: 0, exid: 0)
COORDINATES: (-76.263695,34.969795)
             (-76.261551,34.970985)
             (-76.273766,34.977989)
             (-76.281067,34.985764)
             (-76.283165,34.993248)
             (-76.294685,35.004372)
             (-76.299667,34.999645)
             (-76.296768,34.993778)
             (-76.282333,34.974716)
             (-76.271568,34.970352)
             (-76.263695,34.969795)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
ID: 40
START_NODE: 35
END_NODE: 30
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 32, tile: 0, exid: 0)
LEFT_EDGE: (id: 43, tile: 0, exid: 0)
COORDINATES: (-76.682999,34.950218)
             (-76.682999,35.028908)

ID: 41
START_NODE: 36
END_NODE: 36
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 30, tile: 0, exid: 0)
RIGHT_EDGE: (id: 41, tile: 0, exid: 0)
LEFT_EDGE: (id: 41, tile: 0, exid: 0)
COORDINATES: (-76.207153,34.942383)
             (-76.183647,34.960724)
             (-76.169411,34.974911)
             (-76.170868,34.978432)
             (-76.175880,34.978413)
             (-76.185867,34.972485)
             (-76.207939,34.952972)
             (-76.212181,34.944717)
             (-76.210022,34.942371)
             (-76.207153,34.942383)

ID: 42
START_NODE: 37
END_NODE: 37
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 31, tile: 0, exid: 0)
RIGHT_EDGE: (id: 42, tile: 0, exid: 0)
LEFT_EDGE: (id: 42, tile: 0, exid: 0)
COORDINATES: (-76.319107,34.850117)
             (-76.284920,34.878540)
             (-76.266380,34.889809)
             (-76.264961,34.892174)
             (-76.235756,34.915852)
             (-76.226486,34.922367)
             (-76.225800,34.927666)
             (-76.229385,34.927650)
             (-76.242943,34.920528)
             (-76.280693,34.889160)
             (-76.316330,34.863682)
             (-76.324158,34.855988)
             (-76.321976,34.850693)
             (-76.319107,34.850117)
```

TABLE 98. <u>Edge Table (ECR coverage, tile GJND)</u> - Continued.

```
ID: 43
START_NODE: 39
END_NODE: 35
RIGHT_FACE: (id: 20, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 45, tile: 0, exid: 0)
LEFT_EDGE: (id: 52, tile: 0, exid: 0)
COORDINATES: (-76.682999,34.767570)
             (-76.682999,34.950218)

ID: 44
START_NODE: 40
END_NODE: 40
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 33, tile: 0, exid: 0)
RIGHT_EDGE: (id: 44, tile: 0, exid: 0)
LEFT_EDGE: (id: 44, tile: 0, exid: 0)
COORDINATES: (-76.489479,34.751442)
             (-76.488068,34.754986)
             (-76.489510,34.756752)
             (-76.489525,34.759113)
             (-76.495956,34.757317)
             (-76.495934,34.753185)
             (-76.493774,34.751427)
             (-76.489479,34.751442)

ID: 45
START_NODE: 39
END_NODE: 35
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 20, tile: 0, exid: 0)
RIGHT_EDGE: (id: 40, tile: 0, exid: 0)
LEFT_EDGE: (id: 43, tile: 0, exid: 0)
COORDINATES: (-76.682999,34.767570)
             (-76.677368,34.764057)
             (-76.672287,34.752281)
             (-76.670746,34.735176)
             (-76.671440,34.731636)
             (-76.667831,34.726929)
             (-76.665634,34.718678)
             (-76.672050,34.714218)
             (-76.671318,34.711567)
             (-76.659851,34.708668)
             (-76.654099,34.704556)
             (-76.638336,34.702263)
             (-76.636177,34.701092)
             (-76.629738,34.701710)
             (-76.627617,34.705261)
             (-76.627655,34.711754)
             (-76.631279,34.718819)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.631310,34.724720)
(-76.622841,34.744232)
(-76.620865,34.772556)
(-76.625259,34.789047)
(-76.625298,34.795536)
(-76.621025,34.798500)
(-76.611679,34.792053)
(-76.605652,34.784924)
(-76.605148,34.777340)
(-76.603676,34.771446)
(-76.583794,34.765846)
(-76.578621,34.771553)
(-76.570404,34.785519)
(-76.569206,34.779842)
(-76.568398,34.774342)
(-76.577721,34.763687)
(-76.587784,34.758339)
(-76.595657,34.747887)
(-76.598442,34.734894)
(-76.596970,34.727818)
(-76.589783,34.723721)
(-76.573318,34.722607)
(-76.566170,34.724998)
(-76.559715,34.722668)
(-76.548973,34.722710)
(-76.547539,34.721539)
(-76.540390,34.722748)
(-76.533966,34.726315)
(-76.532120,34.722080)
(-76.526054,34.720448)
(-76.514603,34.720497)
(-76.506218,34.726311)
(-76.506958,34.730442)
(-76.512733,34.738678)
(-76.511177,34.746475)
(-76.518387,34.754704)
(-76.511963,34.757092)
(-76.506233,34.757114)
(-76.503387,34.760078)
(-76.505745,34.767025)
(-76.513336,34.772194)
(-76.507965,34.782169)
(-76.513512,34.788719)
(-76.505699,34.787197)
(-76.499985,34.788990)
(-76.498581,34.794304)
(-76.489838,34.802578)
(-76.485703,34.794949)
(-76.488525,34.787861)
(-76.496338,34.778389)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.498459,34.773075)
(-76.492676,34.764252)
(-76.486938,34.762505)
(-76.481934,34.763119)
(-76.478355,34.764313)
(-76.471970,34.773777)
(-76.464157,34.783836)
(-76.457130,34.805092)
(-76.456467,34.813351)
(-76.461502,34.817455)
(-76.460098,34.822174)
(-76.455803,34.822193)
(-76.450768,34.818680)
(-76.443604,34.816940)
(-76.437187,34.822273)
(-76.435074,34.827587)
(-76.430084,34.830555)
(-76.422897,34.826458)
(-76.421951,34.825706)
(-76.419846,34.833378)
(-76.421341,34.843391)
(-76.418495,34.845760)
(-76.417816,34.852833)
(-76.419319,34.864021)
(-76.413658,34.875240)
(-76.413704,34.882896)
(-76.406082,34.886627)
(-76.403198,34.883698)
(-76.402420,34.872505)
(-76.404533,34.867195)
(-76.401627,34.860138)
(-76.395164,34.856632)
(-76.388725,34.857838)
(-76.377342,34.869080)
(-76.345886,34.876282)
(-76.335190,34.883396)
(-76.332359,34.888706)
(-76.316681,34.900257)
(-76.330330,34.907562)
(-76.342651,34.906929)
(-76.347702,34.912796)
(-76.343277,34.917515)
(-76.340599,34.922245)
(-76.341881,34.923996)
(-76.359062,34.923336)
(-76.361938,34.926266)
(-76.357666,34.929817)
(-76.343369,34.932823)
(-76.341362,34.933479)
(-76.325943,34.928543)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.312988,34.918003)
(-76.305367,34.924191)
(-76.303261,34.931854)
(-76.287323,34.934528)
(-76.287369,34.941593)
(-76.285255,34.947487)
(-76.285301,34.955135)
(-76.298988,34.969204)
(-76.311981,34.986794)
(-76.318474,34.994415)
(-76.320671,35.002640)
(-76.315346,35.003231)
(-76.306534,34.996529)
(-76.304390,34.997124)
(-76.306412,35.004204)
(-76.312180,35.011234)
(-76.346649,35.029320)
(-76.356697,35.033394)
(-76.363136,35.032188)
(-76.362167,35.029179)
(-76.335289,35.016987)
(-76.330238,35.009953)
(-76.330223,35.007599)
(-76.333084,35.007000)
(-76.348885,35.015755)
(-76.354607,35.015732)
(-76.357445,35.010426)
(-76.350945,35.001633)
(-76.327896,34.978199)
(-76.323524,34.965862)
(-76.329674,34.956322)
(-76.337509,34.962757)
(-76.355927,34.964104)
(-76.362930,34.971581)
(-76.371513,34.970367)
(-76.385139,34.974426)
(-76.390144,34.973232)
(-76.387962,34.967941)
(-76.383629,34.960903)
(-76.383598,34.956783)
(-76.392151,34.950272)
(-76.390640,34.937332)
(-76.402084,34.935516)
(-76.406403,34.939030)
(-76.404297,34.946102)
(-76.405762,34.951981)
(-76.405800,34.957863)
(-76.407959,34.960209)
(-76.412964,34.959011)
(-76.423660,34.952492)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
(-76.434357,34.945385)
(-76.440811,34.946533)
(-76.453331,34.939377)
(-76.450050,34.935310)
(-76.452194,34.934715)
(-76.447884,34.932377)
(-76.444954,34.922382)
(-76.451073,34.921135)
(-76.455399,34.925827)
(-76.456863,34.931709)
(-76.464081,34.941685)
(-76.464859,34.951099)
(-76.472786,34.960484)
(-76.466690,34.964077)
(-76.463112,34.964684)
(-76.450218,34.963562)
(-76.437340,34.964794)
(-76.433815,34.974220)
(-76.428833,34.978359)
(-76.431076,34.993645)
(-76.432518,34.995403)
(-76.441284,34.996361)
(-76.444244,35.002426)
(-76.441917,35.010658)
(-76.432686,35.023045)
(-76.423256,35.025822)
(-76.429977,35.029011)
(-76.434792,35.033463)
(-76.430389,35.049927)
(-76.431129,35.054039)
(-76.441177,35.058109)
(-76.466339,35.075634)
(-76.469948,35.079731)
(-76.476379,35.077942)
(-76.478096,35.073994)
(-76.481888,35.068508)
(-76.484741,35.066143)
(-76.478981,35.060291)
(-76.469666,35.059742)
(-76.467499,35.056225)
(-76.459587,35.050381)
(-76.453865,35.050404)
(-76.450264,35.046894)
(-76.460274,35.045677)
(-76.464584,35.047424)
(-76.466698,35.042709)
(-76.462021,35.025864)
(-76.471962,35.032776)
(-76.480659,35.046616)
(-76.486526,35.048683)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

(-76.490799,35.045139)
(-76.493607,35.035721)
(-76.496437,35.029827)
(-76.493523,35.022198)
(-76.485008,35.015472)
(-76.498466,35.012962)
(-76.491188,34.990448)
(-76.480400,34.982258)
(-76.478920,34.975208)
(-76.490913,34.963299)
(-76.488937,34.973988)
(-76.488258,34.980461)
(-76.492828,34.982552)
(-76.510872,34.976868)
(-76.508324,34.983906)
(-76.499786,34.992180)
(-76.502678,34.996872)
(-76.509155,35.002728)
(-76.510620,35.008015)
(-76.519943,35.010326)
(-76.544258,35.006107)
(-76.565689,34.999546)
(-76.579941,34.988899)
(-76.578415,34.973610)
(-76.571190,34.961872)
(-76.556786,34.947811)
(-76.547440,34.941376)
(-76.544487,34.926083)
(-76.545189,34.924316)
(-76.528046,34.900944)
(-76.535095,34.905655)
(-76.550423,34.918095)
(-76.549545,34.934891)
(-76.556755,34.943100)
(-76.572556,34.951275)
(-76.591171,34.951782)
(-76.592758,34.977669)
(-76.602119,34.985863)
(-76.612869,34.988762)
(-76.633476,34.986118)
(-76.637756,34.984333)
(-76.649925,34.983692)
(-76.667061,34.976562)
(-76.680031,34.966709)
(-76.677132,34.961422)
(-76.659119,34.942665)
(-76.646912,34.936245)
(-76.644737,34.931545)
(-76.656868,34.925018)
(-76.658264,34.919712)

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
                (-76.656052,34.908535)
                (-76.658905,34.906757)
                (-76.667511,34.909664)
                (-76.664787,34.932636)
                (-76.664825,34.939114)
                (-76.671310,34.946148)
                (-76.678497,34.950237)
                (-76.682999,34.950218)

ID: 46
START_NODE: 41
END_NODE: 41
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 35, tile: 0, exid: 0)
RIGHT_EDGE: (id: 46, tile: 0, exid: 0)
LEFT_EDGE: (id: 46, tile: 0, exid: 0)
COORDINATES: (-76.608307,34.682129)
                (-76.604324,34.685398)
                (-76.604317,34.689983)
                (-76.608284,34.691299)
                (-76.613846,34.689999)
                (-76.618622,34.685417)
                (-76.615448,34.683449)
                (-76.613861,34.682793)
                (-76.608307,34.682129)

ID: 47
START_NODE: 42
END_NODE: 42
RIGHT_FACE: (id: 34, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 47, tile: 0, exid: 0)
LEFT_EDGE: (id: 47, tile: 0, exid: 0)
COORDINATES: (-76.535240,34.678097)
                (-76 550316,34.685978)
                (-76.571739,34.691906)
                (-76.582062,34.695850)
                (-76.589195,34.701099)
                (-76.593956,34.702415)
                (-76.592354,34.706997)
                (-76.589172,34.709614)
                (-76.580444,34.706982)
                (-76.573288,34.708282)
                (-76.570908,34.708279)
                (-76.569328,34.703037)
                (-76.570137,34.698452)
                (-76.565376,34.696480)
                (-76.561409,34.696476)
                (-76.557434,34.697781)
                (-76.557419,34.701054)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
               (-76.555824,34.704983)
               (-76.548668,34.706284)
               (-76.525635,34.706905)
               (-76.523262,34.703629)
               (-76.520889,34.699692)
               (-76.522484,34.695766)
               (-76.529640,34.695122)
               (-76.539169,34.694477)
               (-76.544731,34.693176)
               (-76.541565,34.689896)
               (-76.534424,34.687920)
               (-76.530457,34.685951)
               (-76.528870,34.683983)
               (-76.528885,34.680706)
               (-76.531265,34.679401)
               (-76.535240,34.678097)

ID: 48
START_NODE: 43
END_NODE: 43
RIGHT_FACE: (id: 37, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 48, tile: 0, exid: 0)
LEFT_EDGE: (id: 48, tile: 0, exid: 0)
COORDINATES: (-76.590057,34.675552)
               (-76.595604,34.678181)
               (-76.594009,34.681454)
               (-76.590034,34.684067)
               (-76.586067,34.682751)
               (-76.583687,34.679474)
               (-76.583702,34.674889)
               (-76.590057,34.675552)

ID: 49
START_NODE: 44
END_NODE: 44
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 36, tile: 0, exid: 0)
RIGHT_EDGE: (id: 49, tile: 0, exid: 0)
LEFT_EDGE: (id: 49, tile: 0, exid: 0)
COORDINATES: (-76.536140,34.637466)
               (-76.531372,34.638115)
               (-76.531357,34.644669)
               (-76.531334,34.653187)
               (-76.534508,34.654503)
               (-76.536118,34.647953)
               (-76.539291,34.649269)
               (-76.546425,34.654522)
               (-76.557541,34.656502)
               (-76.566261,34.662411)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
                (-76.571815,34.665043)
                (-76.580544,34.667019)
                (-76.591652,34.670967)
                (-76.605934,34.676884)
                (-76.618637,34.680832)
                (-76.636887,34.688721)
                (-76.648010,34.687428)
                (-76.647224,34.682838)
                (-76.639290,34.678894)
                (-76.614685,34.673622)
                (-76.600395,34.670322)
                (-76.585320,34.664406)
                (-76.575005,34.659805)
                (-76.565475,34.657825)
                (-76.557541,34.653881)
                (-76.548035,34.644688)
                (-76.542496,34.638130)
                (-76.536140,34.637466)

ID: 50
START_NODE: 38
END_NODE: 38
RIGHT_FACE: (id: 32, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 50, tile: 0, exid: 0)
LEFT_EDGE: (id: 50, tile: 0, exid: 0)
COORDINATES: (-76.332199,34.845810)
                (-76.331497,34.848461)
                (-76.325768,34.848488)
                (-76.323601,34.844372)
                (-76.338539,34.828987)
                (-76.353485,34.815365)
                (-76.364174,34.806477)
                (-76.376984,34.794041)
                (-76.391205,34.778057)
                (-76.411842,34.757915)
                (-76.436028,34.733032)
                (-76.455246,34.714657)
                (-76.497162,34.661934)
                (-76.518440,34.629944)
                (-76.531189,34.606251)
                (-76.533333,34.605652)
                (-76.534767,34.606236)
                (-76.548363,34.605587)
                (-76.548370,34.606770)
                (-76.554108,34.609112)
                (-76.555550,34.610874)
                (-76.554863,34.615017)
                (-76.544121,34.615063)
                (-76.538422,34.619221)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

```
                (-76.538452,34.624542)
                (-76.505081,34.669579)
                (-76.498688,34.677280)
                (-76.487343,34.695629)
                (-76.468178,34.722862)
                (-76.448227,34.737701)
                (-76.444725,34.750698)
                (-76.439766,34.758976)
                (-76.433342,34.762543)
                (-76.424034,34.761402)
                (-76.416183,34.765564)
                (-76.415504,34.771469)
                (-76.413406,34.779732)
                (-76.390610,34.798111)
                (-76.374977,34.817043)
                (-76.360008,34.828304)
                (-76.339333,34.842537)
                (-76.332199,34.845810)

ID: 51
START_NODE: 45
END_NODE: 6
RIGHT_FACE: (id: 1, tile: 0, exid: 0)
LEFT_FACE: (id: 2, tile: 0, exid: 0)
RIGHT_EDGE: (id: 5, tile: 15, exid: 2)
LEFT_EDGE: (id: 52, tile: 0, exid: 0)
COORDINATES: (-75.000000,34.583000)
             (-75.000000,36.000000)

ID: 52
START_NODE: 45
END_NODE: 39
RIGHT_FACE: (id: 2, tile: 0, exid: 0)
LEFT_FACE: (id: 1, tile: 0, exid: 0)
RIGHT_EDGE: (id: 45, tile: 0, exid: 0)
LEFT_EDGE: (id: 51, tile: 15, exid: 4)
COORDINATES: (-75.000000,34.583000)
             (-76.682999,34.583000)
             (-76.682999,34.767570)
```

TABLE 98. Edge Table (ECR coverage, tile GJND) - Continued.

TABLE 99. Edge Bounding Rectangle Table (ECR coverage, tile GJND).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <XMIN> <Minimum X Coordinate>  Type: F Count: 1 KeyType: N
Name: <YMIN> <Minimum Y Coordinate>  Type: F Count: 1 KeyType: N
Name: <XMAX> <Maximum X Coordinate>  Type: F Count: 1 KeyType: N
Name: <YMAX> <Maximum Y Coordinate>  Type: F Count: 1 KeyType: N
```

ID	XMIN	YMIN	XMAX	YMAX
1	-76.682999	36.000000	-75.724709	36.000000
2	-75.724709	36.000000	-75.695251	36.000000
3	-75.695251	36.000000	-75.688568	36.000000
4	-75.688568	36.000000	-75.658348	36.000000
5	-75.658348	36.000000	-75.000000	36.000000
6	-75.724709	35.997810	-75.695251	36.000000
7	-76.682999	35.938274	-76.682999	36.000000
8	-75.944977	35.937912	-75.869843	35.981689
9	-75.734314	35.837349	-75.618668	35.943443
10	-75.595352	35.799721	-75.583893	35.807880
11	-75.688568	35.794704	-75.535164	36.000000
12	-75.562187	35.793606	-75.553482	35.802879
13	-76.682999	35.502422	-76.682999	35.938274
14	-76.682999	35.492249	-76.682999	35.502422
15	-76.682999	35.415573	-76.682999	35.492249
16	-76.077568	35.397129	-76.065498	35.405354
17	-76.682999	35.387768	-76.566322	35.510365
18	-76.410080	35.346630	-76.403633	35.352501
19	-76.682999	35.341675	-76.682999	35.415573
20	-76.368309	35.335648	-76.343666	35.348064
21	-76.682999	35.331284	-75.711357	35.995605
22	-76.288292	35.313805	-76.270271	35.336613
23	-75.732666	35.192982	-75.463051	35.767998
24	-76.682999	35.162468	-76.682999	35.341675
25	-76.682999	35.162270	-76.468460	35.348843
26	-76.537216	35.148808	-76.527184	35.157619
27	-76.682999	35.144127	-76.682999	35.162468
28	-76.002548	35.078228	-75.761215	35.188152
29	-76.051003	35.066784	-76.043846	35.072052
30	-76.438469	35.065533	-76.428391	35.075512
31	-76.059021	35.048023	-76.033531	35.063412
32	-76.682999	35.028908	-76.540024	35.176758
33	-76.682999	35.028908	-76.682999	35.144127
34	-76.123154	35.018051	-76.058327	35.074734
35	-76.116081	35.007931	-76.063728	35.045845
36	-76.156639	34.992039	-76.140251	35.007381

37	-76.155693	34.978249	-76.128166	35.002449
38	-76.266655	34.977451	-76.251602	34.996883
39	-76.299667	34.969795	-76.261551	35.004372
40	-76.682999	34.950218	-76.682999	35.028908
41	-76.212181	34.942371	-76.169411	34.978432
42	-76.324158	34.850117	-76.225800	34.927666
43	-76.682999	34.767570	-76.682999	34.950218
44	-76.495956	34.751427	-76.488068	34.759113
45	-76.682999	34.701092	-76.285255	35.079731
46	-76.618622	34.682129	-76.604317	34.691299
47	-76.593956	34.678097	-76.520889	34.709614
48	-76.595604	34.674889	-76.583687	34.684067
49	-76.648010	34.637466	-76.531334	34.688721
50	-76.555550	34.605587	-76.323601	34.848488
51	-75.000000	34.583000	-75.000000	36.000000
52	-76.682999	34.583000	-75.000000	34.767570

TABLE 99. Edge Bounding Rectangle Table (ECR coverage, tile GJND) - Continued.

TABLE 100. Ring Table (ECR coverage, tile GJND).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <FACE_ID> <Foreign key to face table> Type: I Count: 1 KeyType: N
Name: <START_EDGE> <Foreign Key to Edge Table> Type: I Count: 1 KeyType: N

ID	FACE_ID	START_EDGE
1	1	Null
2	2	1
3	2	8
4	2	9
5	2	10
6	2	12
7	2	16
8	2	18
9	2	20
10	2	22
11	2	23
12	2	26
13	2	28
14	2	29
15	2	30
16	2	31
17	2	34
18	2	35
19	2	36
20	2	37
21	2	38
22	2	39
23	2	41
24	2	42
25	2	44
26	2	46
27	2	47
28	2	48
29	2	49
30	2	50
31	3	2
32	4	4
33	5	13
34	6	8
35	7	9
36	8	10
37	9	12
38	10	23
39	11	15
40	12	16

41	13	18
42	14	24
43	15	20
44	16	22
45	17	28
46	18	32
47	19	26
48	20	43
49	21	30
50	22	34
51	23	29
52	24	31
53	25	35
54	26	36
55	27	39
56	28	37
57	29	38
58	30	41
59	31	42
60	32	50
61	33	44
62	34	47
63	35	46
64	36	49
65	37	48

TABLE 100. Ring Table (ECR coverage, tile GJND) - Continued.

TABLE 101. Face Table (ECR coverage, tile GJND).

Table Definition:
Name: <ID> <Row ID> Type: I Count: 1 KeyType: P
Name: <RING_PTR> <Foreign Key to Ring Table> Type: I Count: 1 KeyType: N

ID	RING_PTR
1	1
2	2
3	31
4	32
5	33
6	34
7	35
8	36
9	37
10	38
11	39
12	40

13	41
14	42
15	43
16	44
17	45
18	46
19	47
20	48
21	49
22	50
23	51
24	52
25	53
26	54
27	55
28	56
29	57
30	58
31	59
32	60
33	61
34	62
35	63
36	64
37	65

TABLE 101. Face Table (ECR coverage, tile GJND) - Continued.

TABLE 102. Face Bounding Rectangle Table (ECR coverage, tile GJND).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <XMIN> <Minimum X Coordinate>  Type: F Count: 1 KeyType: N
Name: <YMIN> <Minimum Y Coordinate>  Type: F Count: 1 KeyType: N
Name: <XMAX> <Maximum X Coordinate>  Type: F Count: 1 KeyType: N
Name: <YMAX> <Maximum Y Coordinate>  Type: F Count: 1 KeyType: N
```

ID	XMIN	YMIN	XMAX	YMAX
1	Null	Null	Null	Null
2	-76.683029	34.583000	-75.000000	36.000000
3	-75.724716	35.997810	-75.695251	36.000000
4	-75.688583	35.794704	-75.535164	36.000000
5	-76.683029	35.331284	-75.711357	35.995605
6	-75.944992	35.937912	-75.869843	35.981689
7	-75.734360	35.837349	-75.618668	35.943443

8	-75.595383	35.799721	-75.583893	35.807880
9	-75.562180	35.793606	-75.553482	35.802879
10	-75.732712	35.192982	-75.463051	35.767998
11	-76.683029	35.387768	-76.566322	35.510365
12	-76.077560	35.397129	-76.065498	35.405354
13	-76.410080	35.346630	-76.403633	35.352501
14	-76.683029	35.162270	-76.468460	35.348843
15	-76.368332	35.335648	-76.343666	35.348064
16	-76.288315	35.313805	-76.270271	35.336613
17	-76.002548	35.078228	-75.761215	35.188152
18	-76.683029	35.028908	-76.540024	35.176758
19	-76.537216	35.148808	-76.527184	35.157619
20	-76.683029	34.701092	-76.285255	35.079731
21	-76.438461	35.065533	-76.428391	35.075512
22	-76.123154	35.018051	-76.058327	35.074734
23	-76.051010	35.066784	-76.043846	35.072052
24	-76.059067	35.048023	-76.033531	35.063412
25	-76.116074	35.007931	-76.063728	35.045845
26	-76.156662	34.992039	-76.140251	35.007381
27	-76.299667	34.969795	-76.261551	35.004372
28	-76.155685	34.978249	-76.128166	35.002449
29	-76.266647	34.977451	-76.251602	34.996883
30	-76.212204	34.942371	-76.169411	34.978432
31	-76.324203	34.850117	-76.225800	34.927666
32	-76.555588	34.605587	-76.323601	34.848488
33	-76.495956	34.751427	-76.488068	34.759113
34	-76.593979	34.678097	-76.520889	34.709614
35	-76.618637	34.682129	-76.604317	34.691299
36	-76.648056	34.637466	-76.531334	34.688721
37	-76.595627	34.674889	-76.583687	34.684067

TABLE 102. Face Bounding Rectangle Table (ECR coverage, tile GJND) - Continued.

TABLE 103. Text Table (ECR coverage, tile GJND).

```
Table Definition:
Name: <ID> <Row ID>  Type: I Count: 1 KeyType: P
Name: <STRING> <Text String>  Type: T Count: -1 KeyType: N
Name: <SHAPE_LINE> <The shape line>  Type: C Count: * KeyType: N


ID: 1
STRING: ALBEMARLE SOUND
SHAPE_LINE: (-76.429352,35.995964)
            (-76.429352,35.995964)
            (-75.773872,36.006542)
```

```
ID: 2
STRING: Nage
SHAPE_LINE: (-75.755089,35.975903)
            (-75.755089,35.975903)
            (-75.672562,35.977703)

ID: 3
STRING: Head
SHAPE_LINE: (-75.722145,35.952892)
            (-75.722145,35.952892)
            (-75.644341,35.952785)

ID: 4
STRING: BODIE ISLAND
SHAPE_LINE: (-75.775040,35.845127)
            (-75.775040,35.845127)
            (-75.596321,35.842068)

ID: 5
STRING: NORTH
SHAPE_LINE: (-76.415779,35.777542)
            (-76.415779,35.777542)
            (-75.873505,35.776794)

ID: 6
STRING: Oregon Inlet
SHAPE_LINE: (-75.708374,35.764370)
            (-75.708374,35.764370)
            (-75.541176,35.766018)

ID: 7
STRING: CAROLINA
SHAPE_LINE: (-76.536469,35.608696)
            (-76.536469,35.608696)
            (-75.671188,35.605579)

ID: 8
STRING: Hatteras Inlet
SHAPE_LINE: (-75.959717,35.268799)
            (-75.959717,35.268799)
            (-75.752419,35.197037)

ID: 9
STRING: CAPE HATTERAS
SHAPE_LINE: (-75.478790,35.246891)
            (-75.478790,35.246891)
            (-75.118042,35.252182)
```

TABLE 103. Text Table (ECR coverage, tile GJND) - Continued.

```
ID: 10
STRING: Ocracoke Inlet
SHAPE_LINE: (-76.221718,35.157097)
            (-76.221718,35.157097)
            (-76.033279,35.087200)

ID: 11
STRING: PAMLICO SOUND
SHAPE_LINE: (-76.372719,35.114758)
            (-76.372719,35.114758)
            (-75.558098,35.572731)

ID: 12
STRING: Portsmouth
SHAPE_LINE: (-76.280914,35.060448)
            (-76.280914,35.060448)
            (-76.097008,35.060192)

ID: 13
STRING: RALEIGH  BAY
SHAPE_LINE: (-76.120888,34.943993)
            (-76.120888,34.943993)
            (-75.726608,35.148674)

ID: 14
STRING: Beaufort
SHAPE_LINE: (-76.650787,34.718575)
            (-76.650787,34.718575)
            (-76.502151,34.757236)

ID: 15
STRING: CAPE LOOKOUT
SHAPE_LINE: (-76.511833,34.661983)
            (-76.511833,34.661983)
            (-76.339241,34.844395)
```

TABLE 103. Text Table (ECR coverage, tile GJND) - Continued.

CONCLUDING MATERIAL

Custodians:	Preparing activity:
DMA - MP	DMA - MP
Air Force - 09	(Project MCGT-0146)
Army - TI	
Navy - NO	

Review activity:
Marine Corps - MC

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

INSTRUCTIONS

1. The preparing activity must complete blocks 1, 2, 3, and 8. In block 1, both the document number and revision letter should be given.
2. The submitter of this form must complete blocks 4, 5, 6, and 7.
3. The preparing activity must provide a reply within 30 days from receipt of the form.

NOTE: This form may not be used to request copies of documents, nor to request waivers, or clarification of requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document(s) or to amend contractual requirements.

I RECOMMEND A CHANGE:	1. DOCUMENT NUMBER MIL-STD-2407	2. DOCUMENT DATE *(YYMMDD)* 960628

3. DOCUMENT TITLE Interface Standard for Vector Product Format

4. NATURE OF CHANGE *(Identify paragraph number and include proposed rewrite, if possible. Attach extra sheets as needed)*

5. REASON FOR RECOMMENDATION

6. SUBMITTER

a. NAME *(Last, First, Middle Initial)*	b. ORGANIZATION	
c. ADDRESS *(Include Zip Code)*	d. TELEPHONE *(Include Area Code)* (1) Commercial (2) AUTOVON *(if applicable)*	7. DATE SUBMITTED *(YYMMDD)*

8. PREPARING ACTIVITY

a. NAME Defense Mapping Agency ATTN: ATIS, ST A-10	b. TELEPHONE *(Include Area Code)* (1) Commercial (703) 275-8509	(2) AUTOVON 235-8509
c. ADDRESS *(Include Zip Code)* 8613 Lee Highway Fairfax, VA 22031-2137	IF YOU DO NOT RECEIVE A REPLY WITHIN 45 DAYS, CONTACT: Defense Quality and Standardization Office 5203 Leesburg Pike, Suite 1403, Falls Church, VA 22041-3466 Telephone (703) 756-2340 AUTOVON 289-2340	

DD Form 1426, OCT 89 *Previous editions are obsolete.* 198-290